WEWoRC 2007

Western European Workshop on Research in Cryptology

Chairs:
Stefan Lucks, Ahmad-Reza Sadeghi, Christopher Wolf

Bochum, Germany, July 4-6, 2007

Conference Record

WEWoRC is jointly organized by

Silver Sponsors

Bronze Sponsors

Thanks

We want to thank Christian Wachsmann and Jamshid Shokrollahi who have helped preparing this conference record. Moreover, we want to thank all authors for their contribution.

Contents

A Privacy Protection Scheme for a Scalable Control Method in Context-dependent Services
Rei Yoshida, Rie Shigetomi, Kazuki Yoshizoe, Akira Otsuka, and Hideki Imai 1
The GPS identification scheme using Frobenius expansions
Waldyr Benits and Steven Galbraith 6
Efficient Assembly Implementation of Dragon, LEX, Salsa20 and Sosemanuk on 8-bit AVR Microcontrollers
Gordon Meiser, Thomas Eisenbarth, Kerstin Lemke-Rust, Christof Paar 11
MPSoC-coupled Hardware Accelerator for Elliptic Curve Cryptography in Network Application Domain
Christoph Puttmann and Jamshid Shokrollahi 16
Finding message pairs conforming to simple SHA-256 characteristics: Work in Progress
Marko Hölbl, Christian Rechberger, Tatjana Welzer 21
XOR-Hash : A Hash Function based on XOR
Stéphane Manuel and Nicolas Sendrier 26
Efficient Hash Collision Search Strategies on Special-Purpose Hardware
Tim Güneysu, Christof Paar, Sven Schäge 31
Group Signatures With Observers Revisited
Thomas Schwarzpaul 36
Modified HB Authentication Protocol
Selwyn Piramuthu, Yu-Ju Tu 41
Cryptography Based on Quadratic Forms: Complexity Considerations
Rupert J. Hartung 45
Lattice-Based Computationally-Efficient Private Information Retrieval Protocol
Carlos Aguilar-Melchor and Philippe Gaborit 50
A McEliece-like Symmetric-Key Cryptosystem based on Quasi-Cyclic Low-Density Parity-Check Codes
Ali Akbar Sobhi Afshar, Taraneh Eghlidos, Mohammad Reza Aref 55
A Three Based Chaotic Maps Block Cipher
Mabrouka Hagui , Hedi Khammari and Bechir Ayeb 61

Toward a Concrete Security Proof of Courtois, Finiasz and Sendrier Signature Scheme
Léonard Dallot 66
HFE - solved or saved?
Jintai Ding, Dieter Schmidt and Fabian Werner 68
AES Cache-timing attacks verified
Joris Plessers 70
A Lightweight Hardware Implementation of the Stream Cipher VEST-4
Timo Gendrullis, Timo Kasper, and Christof Paar 72
Cryptanalysis of the MOR System (extended abstract)
Anja Korsten 74
On the Permutations of KHAZAD
Ralph Wernsdorf 76
Exploring Trade-offs between Area, Performance and Security in HW/SW Co-design of ECC
Caroline Vanderheyden, Junfeng Fan, Kazuo Sakiyama, and Ingrid Verbauwhede 79
Low-weight Joint Window Nonadjacent Form
Wei-Hua He and Fei-Ming Juan 82
BDD-based Cryptanalysis of Hardware-oriented Stream Ciphers
Dirk Stegemann 84
Preimages for Reduced-Round Tiger
Sebastiaan Indesteege 86
Algebraic Attacks on Block Ciphers with Several Plaintext-Ciphertext Pairs
Nicolas T.Courtois and Blandine Debraize 88
Lattice-based partial key exposure attacks on signature schemes
Paul Kubwalo, John Mulagha, and Edward F. Schaefer 90
Template Attacks On Masking: An Interpretation
Benedikt Gierlichs and Kåre Janussen 92
Wibree Security — towards better energy efficiency
Jan-Erik Ekberg 97
A new Security Model and a Generic Construction for CCA2-Secure Certificateless Cryptosystems
Ewan Fleischmann 103
Encryption Based Trust Negotiation
Dalia Khader 108
A Threshold Scheme with Disenrollment (extended abstract)
Michael Beiter 115

A New Threshold Audio Secret Sharing Scheme
Mohammad Ehdaie, Taraneh Eghlidos, Mohammad Reza Aref 119
Attacks on the Stream Ciphers TPy6 and Py6andDesign of New Ciphers TPy6-A and TPy6-B
Gautham Sekar, Souradyuti Paul and Bart Preneel 124
The Impact of Entropy Loss caused by Random Functions
Andrea Röck 130
Cryptanalysis of Achterbahn-128/80 with a new keystream limitation
María Naya-Plasencia 136
Using Two-Steps Hash Function to Support Trustworthy Signing in XML Signature
Sebastian Gajek, Lijun Liao and Jörg Schwenk 141
Browser-based Authentication Protocols for Naive Users
Sebastian Gajek, Mark Manulis, Ahmad-Reza Sadeghi and Jörg Schwenk 147
A Novel Impossible Differential Cryptanalysis of AES
Behnam Bahrak and Mohammad Reza Aref 152
Design Criteria for Key Mixture Functions of Block Ciphers
M. Tayarani Najaran, T. Eghlidos and M. R. Aref 157
New Key Schedule Prototype With Provable Security
M. Tayarani Najaran, T. Eghlidos and M. R. Aref 163

We have listed all papers submitted and accepted for WEWoRC. Unfortunately, some authors had visa problems and hence could not attend the workshop. They have been listed in this conference record nevertheless.

The responsibility for the correctness of the results presented lies solely with the corresponding authors. In particular, the papers collected in this conference record have not been peer reviewed.

A Privacy Protection Scheme for a Scalable Control Method in Context-dependent Services

Rei Yoshida[1], Rie Shigetomi[2], Kazuki Yoshizoe[1], Akira Otsuka[2], and Hideki Imai[12]

[1] Imai Lab., Dept. of EECE, Faculty of Science and Engineering,
Chuo University, Tokyo, Japan
{n57038@educ.kc,yoshizoe@tamacc}.chuo-u.ac.jp
[2] Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
http://www.rcis.aist.go.jp/index-en.html
{rie-shigetomi,a-otsuka,h-imai}@aist.go.jp

Abstract

Context-dependent services are triggered when the context satisfies an execution condition. To deliver these services, user's contexts have to be acquired by terminals such as GPS. However this service has problem of efficiency that it is necessary to collect as many contexts as possible to provide services appropriately, and problem of privacy that the all data concentrates on one place. Many previous works have tackled to only either one of the two problems. In this paper, we propose a scheme to protect users' privacy while keeping efficiency by using Randomized Response Technique.

Keywords. Context-dependent Service, Privacy Protection, Randomized Response Technique

1 Introduction

Context is any information that can be used to characterize the situation of entities (i.e. a person, a place, or an object), or that are relevant to the interaction between a user and an application[4]. A context-dependent service changes its behavior according to the context of the user or his/her surroundings, and it assists his/her activity in the real world. The development of devices such as GPS, mobile phones and non-contact type IC cards makes the service possible.

One example of context-dependant service is restaurant recommendation service. This service automatically shows the restaurant recommendation on the display of the user's terminal. The contexts to be monitored are the user's distance from restaurant and the number of available tables in it. The recommended restaurant should be close enough and must have unoccupied seat. If the user registers his favor beforehand, the service may be able to recommend the restaurant corresponding to the favor. If the recommended restaurant distributes coupons, the service can provide its coupon. Other examples are, management of user's presence in the office, children surveillance by using mobile phone's GPS, 24 hours health-care at home, and etc..

This kind of service system constitutes of context acquisition terminals, servers and users. Context acquisition terminals are connected with servers via a network: the terminals are such as mobile phones with GPS or sensor nodes. The server collects values which the terminals acquired via the network. The server determines whether it is an appropriate time to execute the service by referring to the service's execution conditions. If the execution conditions are satisfied as a result of the determination, the service is provided to the user.

This service have become popular nowadays for providing convenient and efficient service for the customer. However, two problems arise when we control context-dependent services, one is efficiency and the other is user's privacy.

The first problem is that the server has to monitor as many contexts as possible in real time, to make the service execution of each service appropriate. This will cause problems such as excessive computation workload of the server and excessive bandwidth consumption of network. The user's privacy becomes a problem because all contexts are gathered to one place such as the user's name, the user's location and provided service to the user.

1.1 Related Work

That problem of efficiency is already studied in several previous work[3, 5, 6, 7]. The simplest solution is periodical communication method. However, the periodical method with low frequency would miss many appropriate chances to execute services. Prabhakar et al. proposed in [5], a Q-index approach, in which each mobile object is assigned a safe region. Query is only necessary when the objects cross the boundaries of safe regions. With this method, the number of unnecessary location update can be reduced. Hu et al. proposed an efficient algorithm for safe region computation in [7]. Cai et al. proposed in [6] that the objects monitor the query region. Part of server's computation workload is distributed to the objects.

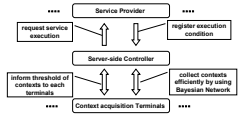


Figure 1: Architecture of BN method

Also, in [3], Uchida et al. suggested a system based on Bayesian Network for calculating execution probability. We call the method 'BN' in this paper (Fig.1). The BN method has a server-side controller in addition to a server and context acquisition terminals. A server-side controller collects context values from the acquisition terminals at appropriate times that are determined by considering the probability of the service execution. When acquired contexts satisfied the service execution condition, the controller requests service execution to the server. In addition, the controller informs each context acquisition terminal of the threshold of contexts. When the context value exceeds the threshold, the acquisition terminals send its value to the controller voluntarily. As mentioned above, the BN method reduce unnecessary communication by configuring cycles for collecting contexts, and using thresholds for each terminals. The simulation of the BN method shows that the BN method is ten times more efficient than the existing periodical communication method.

However, these approaches including BN method still has a possibility of ruining user's privacy because all contexts are gathered to one entity.

The second problem of privacy has been studied by a few previous work[11, 12, 13]. Context Toolkit Project[11] is one of the first methods to challenging privacy problem on context-aware system. The Context Toolkit consists of context widgets, interpreters and aggregators. A special widget, which acts as a gateway between other widgets and application, provides basic access control for privacy protection. Context Broker Architecture (CoBrA) [12] uses the Web Ontology Language OWL for supporting context reasoning. The central entity of this architecture is the server called context broker. To protect user's privacy, the broker enforces the privacy policies

defined by users. However, these methods do not consider efficiency. To provide context-dependent service in real world, we have to address the problems of efficiency and privacy. Roussaki et al. challenged the two problems in [13]. Their method trusts the service provider and aims to protect user's privacy data from third party. To protect privacy, the user's terminal encapsulate and segmentalize his context information with pseudonyms. Also the database servers are distributed based on location. According to the tendency that most users request the information related to their current location, authors claimed the system meets efficiency. However this method requires frequent operations on each local database server.

In this paper, we propose a system, satisfying both privacy protection and efficiency without trusting the service provider. For considering efficiency, we adopted BN method which uses Bayesian Network in our system because this approach is the most efficient method currently known. This approach could be well combined with Randomized Response Technique(RRT)[8] because these schemes could be used in learning part. Namely, our method uses RRT to train Bayesian Network while protecting privacy and keeping efficiency.

2 Proposed System

In this section, we will explain our proposed system. Fig.2 shows the architectures of the method.

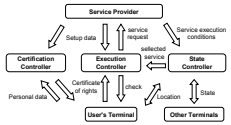


Figure 2: Architecture of proposed method

The entities are service providers $\mathcal{SP}$, context acquisition terminals $\mathcal{U}$, the certification controller $\mathcal{C}_c$, the execution controller $\mathcal{C}_e$ and the state controller $\mathcal{C}_s$. $\mathcal{C}_c$ is entrusted to the trusted third certification authority. $\mathcal{C}_s$ is the server to comprehend whole states such as BN method's controller. $\mathcal{C}_e$ is a controller for verification of certification and determination of service execution to each users such as applications provided from $\mathcal{SP}$. The requirement of privacy for our method is to separate each of user's identity, user's location history and user's service history. To protect privacy while keeping efficiency, we use simple segmentation that $\mathcal{C}_c$, $\mathcal{C}_s$ and $\mathcal{C}_e$ have the responsibility in respectively.

The flow of our system is shown below.

Firstly, $\mathcal{SP}$ shows the list of relation authorities to services to $\mathcal{C}_e$ $\mathcal{SP}$ shows the information to set up the Bayesian Network such as past service execution results to $\mathcal{C}_s$.

Next, $\mathcal{C}_c$ issues certifications which service the user can be provided. $\mathcal{C}_s$ collects C' which is context without personal data at appropriate times and informs threshold of contexts to each terminals such as BN method's controller. When $\mathcal{C}_s$ could check the execution conditions are satisfied, $\mathcal{C}_s$ informs that the service S is satisfied all execution condition to $\mathcal{C}_e$.

Then, $\mathcal{C}_e$ gets the certification issued by $\mathcal{C}_c$ as the context of $\mathcal{U}$'s authority. When $\mathcal{U}$ has the authority to be provided the S, $\mathcal{C}_e$ require the service execution of $\mathcal{SP}$. $\mathcal{SP}$ provide the service to $\mathcal{U}$.

Finally, $\mathcal{C}_e$ must inform that whether $\mathcal{U}$ could be provided service or not, to $\mathcal{C}_s$ for updating the Bayesian Network. Because the Bayesian Network uses the past results for prediction and the accuracy of prediction is improved by learning new service execution results. However, if a particular service may be provided only a particular user in a particular location, it's natural that SP can know a certain user's location in service execution time. Therefore it must disconnect service execution result with user's location history to protect user's privacy. To achieve this contradiction, we propose the use of RRT that is probabilistic scheme as well as Bayesian Network.

RRT is traditionally proposed for polls relating sensitive issues such as drug abuse, criminal record[8]. The underlying idea is for respondent to randomize each response according to a certain, and known probability distribution. When evaluating all the answers of the poll, these lies become statistically insignificant given a large enough sample. Respondents $\mathcal{R}$ answer the question truthfully with some probability $P_{ct} > 1/2$, while with a fixed and known probability $1 - P_{ct}$ they lie. Assume that interviewers $\mathcal{I}$ can not know the answer whether truth or lie. Therefore the $\mathcal{R}$'s privacy is protected because the $\mathcal{I}$ can not connect the answer and the $\mathcal{R}$. Over all, if the P_{ct} is stabilized and $\mathcal{I}$ get enough number of answers, $\mathcal{I}$ can statistically true information with easy calculation.

Let π_A be the true proportion of the population that whose type is t to all answers N. A probabilistic algorithm that, given answers $t'_1, \ldots, t'_N$ and probability of true P_{ct}, and outputs the true information π_A by using simple calculation: $\pi_A = (P_A - (1 - P_{ct}))/(2P_{ct} - 1)$.

There are some methods for implementation of RRT[9, 10]. We use one of the methods with oblivious transfer[9]. By using this method, $\mathcal{R}$ can not know which answer $\mathcal{I}$ use and $\mathcal{I}$ can not know $\mathcal{R}$'s real answer. In addition, only when $\mathcal{R}$ does a correct input that has true probability of lies, $\mathcal{R}$ can be provided services. If $\mathcal{R}$ incorrect input such as the lie 100%, $\mathcal{I}$ can notice the wrong. Therefore, our method uses RRT to update Bayesian Network with the service execution result while protecting privacy. Also, RRT keeps efficiency because it requires only simple calculation.

3 Conclusion & Discussion

We proposed a method which solves both efficiency and privacy protection in context-dependent service.

For efficiency, we rely on the state controller's efficiency. In this paper, we used BN method which is described in [3] because it is most efficient method currently known. The number of communication has increased by 1 time to check the user's authority at $\mathcal{C}_e$, but our method is much more efficient than existing periodical method.

For the problem of privacy, we use three controllers to separate user's identity, user's location history and user's service history: $\mathcal{C}_c$ can not know the reality of the user's location and service histories, $\mathcal{C}_s$ can not know the user's personal data and can not learn the service execution history without RRT, and, $\mathcal{C}_e$ can know various data because it is user's application. $\mathcal{SP}$ is able to know only information about the users who have a true authority to receive the service. Accordingly the user's privacy protection is achieved. This simple segmentation is does not affect for efficiency because these functions are originally separated, but has one problem that the Bayesian Network must be updated BN with the provided service results. We propose using RRT between $\mathcal{C}_s$ and $\mathcal{C}_e$ which can update while protecting privacy and keeping efficiency. In this paper, the amount of calculation increases due to using oblivious transfer. However, there is little problem because the calculation could be postponed to midnight, for example, to alleviate computational workload.

References

[1] R. Yoshida, R. Shigetomi, and H. ImaiC "A Privacy Protection Scheme for a Scalable Control Method in Context-dependent Services"(In Japanese), Proceeding of the 29th Symposium on Information Theory and Its Applications, pp.327-330, 2006.

[2] R. Yoshida, R. Shigetomi, K. Yoshizoe, A. Otsuka, and H.ImaiC "A Study of the Privacy Protection Scheme for a Scalable Control Method in Context-dependent Services(In Japanese)"C Proceedings of the 2007 Symposium on Cryptography and Information Security.

[3] W. Uchida, H. Kasai, and S. Kurakake, "A Scalable Execution Control Method for Context-dependent Services", Proc. IEEE Int'l Conf. on Pervasive Services (ICPS'06), pp.121-130.

[4] A. K. Dey, G. D. Abowd, and D. Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", HCI Journal, Vol. 16 (2-4), pp.97-166, 2001.

[5] S. Prabhakar, Y. Xia, and D. Kalashnikov, W. G. Aref, and S. Hambrusch, "Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects", IEEE Transactions on Computers, 15(10):1124-1140, October 2002.

[6] Y. Cai, K. A. Hua, and G. Cao, "Processing Range-Monitoring Queries on Heterogeneous Mobile Objects", Proc. 2004 IEEE Int'l Conf. on Mobile Data Management(MDM'04), pp.27-38, 2004.

[7] H. Hu, J. Xu, and D. L. Lee, "A Generic Framework for Monitering Continuous Spatial Queries over Moving Objects", Proc. 2005 ACM SIGMOD Int'l Conf. on Management of Data, pp.479-490, 2005.

[8] S. L. Warner, "Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias", Journal of the American Statistical Association, 60(309):63-69, March 1965.

[9] A. Ambainis, M. Jakobsson, and H. Lipmaa, "Cryptographic Randomized Response Technique", Proceedings of PKC '04. LNCS 2947. Springer-Verlag, 2004. pp.425–438.

[10] H. Kikuchi, J. Akiyama, G. Nakamura, and H. Gobioff, "Stochastic Voting Protocol To Protect Voters Privacy", In 1999 IEEE Workshop on Internet Applications, pages 103-111, July 26-27 1999.

[11] D. Salber, A. K. Dey, G. D. Abowd, "The Context Toolkit: Aiding the Development of Context-Enabled Applications", Conference on Human Factors in Computing Systems 1999, May 1999.

[12] H. Chen, T. Finin, A. Joshi, L. Kagal, "Intelligent Agents Meet the Semantics Web in Smart Spaces", IEEE Internet Computing, Vol.8, No.6, Nov.-Dec. 2004, pp.69-79.

[13] I. Roussaki, M. Strimpakou, C. Pils, N. Kalatzis, M. Neubauer,C. Hauser, M. Anagnostou, "Privacy-Aware Modelling and Distribution of Context Information in Pervasive Service Provision", Proc. IEEE Int'l Conf. on Pervasive Services (ICPS'06), pp.150-160.

[14] IST Daidalos Research, URL: http://www.ist-daidalos.org/

The GPS identification scheme using Frobenius expansions

Waldyr Benits and Steven Galbraith

Royal Holloway University of London
www.rhul.ac.uk
[w.benits-junior,steven.galbraith]@rhul.ac.uk

Abstract

The Girault-Poupard-Stern (GPS) identification scheme is designed for public key cryptography on very restricted devices. We propose the use of GPS with random τ-adic expansions representing integers, allowing a much faster scalar multiplication in the offline steps of the scheme. We also explain how this may lead to shorter GPS parameters.

Keywords. Elliptic Curves, Frobenius expansions, GPS identification scheme.

1 Introduction

GPS is an excellent solution for a fast, secure and cheap identification scheme, proposed by Girault [Gir92] and proved secure by Poupard and Stern [PS98]. The three-move protocol is based on Schnorr scheme [C.P90] but does not use modulus calculation in the online step, and therefore, the computational cost of identification is reduced.

In this paper we generalise GPS scheme to elliptic curves and improve the offline steps of the protocol (namely, the Public key generation, the computation of the commitment and the final verification) by speeding up the scalar multiplications. Such efficient scalar multiplication can be achieved, according to Solinas [Sol97], when we represent the scalar as a Frobenius (*aka* τ-adic) expansion instead of its common binary representation. The algorithm for scalar multiplication on elliptic curves using Frobenius expansions runs 50% faster than the well-known double-and-add algorithm.

2 The original GPS scheme

In the standard GPS [Gir92, PS98] we have a prover that holds a private key s and a public key $I = g^{-s} \bmod n$, where n is a RSA modulus and $g \in \mathbb{Z}_n^*$, preferably of maximal order modulo n. In the three step protocol, the prover generates a random r, which she keeps secret, computes the **commitment** $X = g^r \bmod n$ and sends X to the verifier. The verifier, then, generates a random **challenge** c and sends it to the prover. After receiving c, the prover computes the **response** $y = r + cs$ (without modulus reduction) and sends y to the verifier. The computation of y is the only online step. Finally, the verifier checks if $X = g^y I^c$ to be convinced that the prover is in possession of a valid key pair. If $X \neq g^y I^c$, the verifier rejects the proof. The sizes of the parameters and the security analysis of the original GPS can be found in [Gir92, PS98]. An improvement to the original scheme, when only one (online) single addition is performed, can be found in [GL04].

It is easy to generalise the standard GPS scheme to elliptic curves. First we take a public point P and generate the key pair $\{s, I = [-s]P\}$. Then, the prover picks a random integer r, computes the commitment $X = [r]P$ and sends X to verifier, which picks a random integer c and returns it to prover. The answer step is exactly equal to the standard GPS ($y = r + sc$) and the verification becomes to check if $X = [y]P + [c]I$.

3 Koblitz Curves and Frobenius expansions

In this section, we briefly review some properties of Frobenius endomorphism on Koblitz elliptic curves [Kob92] and how we can obtain an integer multiple of a given point in that curve much faster, using Frobenius expansions. Let $E_a : y^2 + xy = x^3 + ax^2 + 1$, with $a \in \{0, 1\}$ be an elliptic curve defined over $\mathbb{F}_2$. Let $E_a(\mathbb{F}_{2^d})$ be the group of $\mathbb{F}_{2^d}$-rational points on E_a. Let $\mu = (-1)^{1-a}$. We need d to be a prime and $\#E_a(\mathbb{F}_{2^d})$ to be a nearly prime. In such curves, defined over $\mathbb{F}_{2^d}$, if $P = (x, y) \in E_a(\mathbb{F}_{2^d})$, so $Q = (x^2, y^2)$ also belongs to the same curve. We can, therefore, define the Frobenius endomorphism as:

$$\begin{array}{rccc} \tau: & E_a(\mathbb{F}_{2^d}) & \to & E_a(\mathbb{F}_{2^d}) \\ & (x, y) & \mapsto & (x^2, y^2) \\ & \mathcal{O} & \mapsto & \mathcal{O} \end{array}$$

Fact. Every integer n can be represented as a Frobenius expansion (see [Kob92]):

$$n = \sum_{i=0}^{\mathcal{N}} n_i \tau^i, \qquad n_i \in \{-1, 0, 1\}$$

So, instead of computing $[n]P$, one can compute $\sum_{i=0}^{\mathcal{N}} n_i \tau^i(P)$.

τ satisfies: $\tau^2 - [\mu]\tau + [2] = 0$.
So,

$$[2]P = [\mu]\tau(P) - \tau(\tau(P))$$

.

Which means that doubling a point (or any scalar multiplication by powers of 2) can be performed using the Frobenius endomorphism, giving, in practice, a improvement of up to 50% compared with the standard double-and-add scalar multiplication [Mül98].

4 The new idea

In the GPS scheme for elliptic curves, instead of using random integers for the private key s, the value r used to compute the commitment and the challenge c, we will use random τ-adic expansions. We emphasize that such modification is not merely to pick a random integer and convert it to a τ-adic expansion in order to accelerate the multiplication steps. Instead, we pick a random τ-adic expansion without worrying which integer it represents.

We only need to care when defining the length of the τ-adic expansion, such that both the DLP (giving $[-s]P$ find s and giving $[r]P$, find r) be hard and also the probability of cheating be negligible.

The advantage of using random τ-adic expansions is that possibly we can reduce the parameters sizes, at the same security level, since the number of possibilities when we have three options for each coefficient ($\{-1, 0, 1\}$) is significantly greater than when we have the standard two options ($\{0, 1\}$).

By using τ-adic expansions, we will get much faster computations for $I = [-s]P$, $X = [r]P$ and, in the verification step, for $X = [y]P + [c]I$. Figure 1 shows the GPS scheme for elliptic curves with τ-adic expansions.[1]

[1] We represent an element x picked at random from a set $\mathcal{X}$ by: $x \xleftarrow{r} \mathcal{X}$.

Public Parameters:		$\mathcal{S}$ (length of s), $\mathcal{R}$ (length of r), $\mathcal{C}$ (length of c), P
Private key:		$s \xleftarrow{r} s_0 + s_1\tau + s_2\tau^2 + \ldots + s_{\mathcal{S}-1}\tau^{\mathcal{S}-1}$, $s_i \in \{0, \pm 1\}$
Public key:		$I = [-s]P$
PROVER		VERIFIER
$r \xleftarrow{r} r_0 + r_1\tau + r_2\tau^2 + \ldots + r_{\mathcal{R}-1}\tau^{\mathcal{R}-1}$, $r_i \in \{0, \pm 1\}$		
Commitment $X = [r]P$	$\xrightarrow{X}$	
		Challenge
	$\xleftarrow{c}$	$c \xleftarrow{r} c_0 + c_1\tau + \ldots + c_{\mathcal{C}-1}\tau^{\mathcal{C}-1}$, $c_i \in \{0, \pm 1\}$
Response $y = r + cs$	$\xrightarrow{y}$	
		$X \stackrel{?}{=} [y]P + [c]I$ If "YES" proof is accepted, otherwise, it is rejected.

Figure 1: GPS

5 Security Issues

We need to define the sizes of the random τ-adic expansions such that the scheme is secure. In this paper, we present a brief description of the security issues. More details can be found in the full version of our paper.

5.1 Size of the Private Key

If the private key s is represented as a τ-adic expansion $s = s_0 + s_1\tau + s_2\tau^2 + \cdots + s_{\mathcal{S}-1}\tau^{\mathcal{S}-1}$, we find s by solving the Discrete Log of $I = [-s]P$. We need $\mathcal{O}(3^{\mathcal{S}})$ steps to find s by exhaustive search (i.e. $\mathcal{S} > 50$ for a security level $> 2^{80}$).

An important aspect to consider is that, when we represent an integer as a τ-adic expansion, we have more than one τ-adic representation (of degree less than some bound) for the same integer (unless we use another type of representation, known as τ-adic NAF, or TNAF for short, which is unique for any integer, but as TNAF expansions are longer than τ-adic, we leave this discussion for the full version). Such fact might lead to an improved Baby-Step-Giant-Step (BSGS) algorithm ([Coh93]) which runs only on distinct equivalent classes of τ-adic expansions.

Hence, it is an important open problem to consider BSGS algorithms in this setting. If we try to find s by using BSGS algorithms, without searching only on distinct equivalent classes of τ-adic expansions, we need $\mathcal{S} > 100$ for the same security level.

It is also important to consider whether low-memory algorithms are available to obtain the private key. If low-memory algorithms are not available, then it may be possible to use shorter private keys than the original GPS scheme.

We proceed with the security analysis for the challenge c, since the same analysis for s applies to r (actually, the analysis of the size of r is more complex, because we have to consider the

zero-knowledge properties, but we leave such analysis to the extended version).

5.2 Size of the Challenge

As c is transmitted clear from verifier, it does not depend on the hardness of the discrete log. However, if an attacker who does not know the secret key can predict c, she can pick a random y, then compute $X = [y]P + [c]I$, send X to Verifier. After receiving c', if $c' = c$, she answers with y and the Verifier will accept. We call *probability of cheating* the probability of a dishonest Prover being successfully accepted by the Verifier.

Therefore, we need c to be large enough such that the probability of cheating be negligible. In practice, we need more than 2^{40} possibilities for c.

If c is represented as a τ-adic expansion $c_0 + c_1\tau + \cdots + c_{\mathcal{C}-1}\tau^{\mathcal{C}-1}$, we need $\mathcal{C} > 25$ (i.e. $3^{\mathcal{C}} > 2^{40}$). In this case, the degree of y will be about 125, but the cost of the online step $y = r + cs$ will be high (at least, when compared to a usual integer multiplication, even if we need larger s and c when dealing with binary representations).

However, to speed up $c \times s$ we can use, for example, a challenge c with at least $\mathcal{S} - 1$ zero coefficients between each pair of non-zero coefficients (see [GL04] for details). If we use such c, before defining the size of c, we need to know how many τ-adic expansions with degree less than $\mathcal{C}$, hamming weight $\leqslant h$ and at least $\mathcal{S} - 1$ zero coefficients between each pair of non-zero coefficients exist, and we can prove that this amount is:

$$Z_{\mathcal{C},\mathcal{S}} = \sum_{h=0}^{\frac{\mathcal{C}+\mathcal{S}-1}{\mathcal{S}}} 2^h \left[\binom{\mathcal{C} - h(\mathcal{S}-1)}{h} + \sum_{i=1}^{\mathcal{S}-1} \binom{\mathcal{C} - i - (h-1)(\mathcal{S}-1)}{h-1} \right] \quad (1)$$

We can see that if we have a 101 degree polynomial s, we need a 660 degree polynomial c for a probability of cheating greater than $\frac{1}{2^{40}}$ and then we end up with a 760 degree polynomial y. It is important to note that each coefficient can be either -1, 0 or 1, which means that we need at least $n \log_2 3$ bits to represent a degree n polynomial. Thus, we need approximately 160, 1046 and 1204 bits to represent respectively s, c and y. Possibly such values can be reduced if low memory algorithms to compute discrete logs are not available in this setting.

Now we have a fast online computation of $c \times s$, at the cost of a very large y to be transmitted, which can be a serious drawback due to memory restrictions. Although c is also very large, it has very few non-zero coefficients, and therefore, the verifier can transmit c by using only the non-zero positions, but as y is expected to be dense, the same cannot be done when sending y.

Our next step will be to implement the GPS using τ-adic expansions in smart cards, to verify the performance and possibly find a tradeoff between time and memory space.

6 Conclusion

We presented the GPS identification scheme using τ-adic expansions to represent integers. Such representation allows a faster scalar multiplication, and therefore, reduces the running times of the offline steps. We also pointed the possibility of using shorter parameters than the original GPS as long as low memory algorithms to compute discrete logs are not available in this setting.

References

[Coh93] Henri Cohen. *A course in computational algebraic number theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[C.P90] C.P.Schnorr. Efficient identification and signatures for smart cards. volume Vol.435 of *Lecture Notes in Computer Science*, pages pp.235–251. Springer Verlag, 1990.

[Gir92] M. Girault. Self-certified public keys. In Springer-Verlag, editor, *Advances in Cryptology – EUROCRYPT 1991*, volume volume 547 of *LNCS*, pages pages 490–497. Springer-Verlag, 1992.

[GL04] Marc Girault and David Lefranc. Public key authentication with one (online) single addition. In *CHES*, pages 413–427, 2004.

[Kob92] Neal Koblitz. Cm-curves with good cryptographic properties. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 279–287, London, UK, 1992. Springer-Verlag.

[Mül98] Volker Müller. Fast multiplication on elliptic curves over small fields of characteristic two. *J. Cryptology*, 11(4):219–234, 1998.

[PS98] Guillaume Poupard and Jacques Stern. Security analysis of a practical "on the fly" authentication and signature generation. *Lecture Notes in Computer Science*, 1403:422–436, 1998.

[Sol97] Jerome A. Solinas. An improved algorithm for arithmetic on a family of elliptic curves. In *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 357–371, London, UK, 1997. Springer-Verlag.

[Sti02] D. R. Stinson. Some baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. *Math. Comput.*, 71(237):379–391, 2002.

Efficient Assembly Implementation of Dragon, LEX, Salsa20 and Sosemanuk on 8-bit AVR Microcontrollers

Gordon Meiser, Thomas Eisenbarth, Kerstin Lemke-Rust, Christof Paar

Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
http://www.crypto.rub.de
{gordon.meiser,eisenbarth,lemke,cpaar}@crypto.rub.de

Abstract

This work is motivated by the question of how efficient modern stream ciphers in the focus of eSTREAM Profile I (Phase 2) can be implemented on small embedded microcontrollers. In response to this question, we present the first implementation results for Dragon, LEX, Salsa20 and Sosemanuk on 8-bit microcontrollers. We implemented these stream ciphers in Assembly to tap the full potential of an embedded implementation. Our efficiency metrics are performance of keystream generation, key setup, and IV setup, and memory usage in flash and SRAM, since microcontrollers are usually strongly constrained in memory resources. Regarding encryption speed, all stream ciphers turned out to outperform AES. In terms of memory needs, Salsa20 and LEX are almost as compact as AES. When considering a time-memory tradeoff metric, LEX and Salsa20 yield significantly better results than AES.

Keywords. stream cipher, software performance, AVR microcontroller, embedded security, eSTREAM, Dragon, LEX, Salsa20, Sosemanuk, AES.

1 Introduction

In 2005, the European Network of Excellence in Cryptology (ECRYPT) launched a call for stream cipher primitives [1] to identify new stream ciphers suitable for widespread adoption that may also serve as an alternative for the AES [5]. Profile I of this call asked for stream ciphers for software applications with high throughput requirements, while Profile II aims at identifying stream ciphers suitable for hardware applications with restricted resources such as limited storage, gate count, or power consumption.

The original call says that performance benchmarking for Profile I "may include 8-bit processors (as found in inexpensive smart cards), 32-bit processors (e.g., the Pentium family) to the modern 64-bit processors". However, the current testing framework of the eSTREAM project [3, 8, 2] exclusively targets general-purpose 32-bit and 64-bit CPUs for Profile I candidates. Given the great importance of small embedded controllers in the real world (the market share of embedded processors is more than 99%), we feel that such an evaluation is of value.

This paper is driven by the question of how efficient candidates in the focus of Profile I in Phase 2 can be implemented in Assembly on small 8-bit embedded microprocessors. Small 8-bit microprocessors are constrained in resources such as flash memory and RAM. Besides throughput, efficiency has an additional meaning in this context: resources needed by an implementation of a stream cipher should be kept small, since embedded applications are very often cost constrained. In fact, in many situations cost (given by memory consumption) is more crucial than throughput, in particular since many embedded applications only crypt small payloads.

Though an 8-bit microprocessor may not be the mainstream target platform of Profile I we are confident that there is also a wide public and industrial interest in finding out whether candidates of Profile I can also serve as possible secure alternative of AES on small 8-bit embedded

microprocessors. 8-bit embedded microprocessors are widely used in various applications, including smart cards, household appliances, industrial control, and many more systems. Modern cars, for instance, are equipped with more than fifty microcontrollers. In embedded systems, cryptography is often needed for authentication, secure messaging, and software download.

In this work, we evaluate performance aspects of stream ciphers that are in the focus of Profile 1 in Phase 2 and not broken, yet. In detail, this work covers Dragon [9], HC-128 [10], LEX [7], Salsa20 [6], and Sosemanuk [11]. All these ciphers have been recently advanced to Phase 3 of the eSTREAM project [12].

Table 1: Characteristic sizes of the focused ciphers

Cipher	Key Size [bits]	IV Size [bits]	State Size [bits]
Dragon	128	128	1088
HC-128	128	128	32768
LEX	128	128	256
Salsa20	128	128	512
Sosemanuk	128	128	384

For the first evaluation on the suitability of the Profile I candidates, the C code implementation provided by the designers was ported to an 8-bit AVR microcontroller (for further details see [16]). We did not implement HC-128 in Assembly because its huge consumption of SRAM memory avoids the implementation on any small AVR device. Hence HC-128 is assessed to be not suitable for small embedded microcontrollers.

In our second stage we implemented Dragon, LEX, Salsa20 and Sosemanuk in Assembly to reach the full potential of an embedded implementation for each cipher. The metrics of the stream cipher are compared with an efficient AES implementation in Assembly. Our comparison metric includes (i) throughput of keystream, (ii) time needed for key setup, (iii) time needed for IV setup, (iv) memory allocation in flash (program code), and (v) memory allocation in SRAM (variables).

2 AVR Microcontroller Family

AVR microprocessors are a family of 8-bit RISC microcontrollers. The individual device classes differ in SRAM and flash memory size, as listed in Table 2. Its memory is organized as a Harvard architecture with a 16-bit word program memory and an 8-bit word data memory. The devices of the ATmega series have 32 general purpose registers of 8-bit word size. Most of the microcontroller's instructions are one-cycle. All of the microcontrollers listed in Table 2 can be clocked at up to 16 MHz. Further information belonging to the devices of the ATmega series from Atmel can be found at [15].

Due to its easy usage, its low power consumption and its comparatively low price, the AVR microcontrollers have reached a high popularity in embedded system design.

3 Implementation Results, Assembly Language

This section provides the results on efficiency of the implementation in Assembly. We used an Assembly implementation of the AES cipher [14] to be able to compare our Assembly implementations of Dragon, LEX, Salsa20 and Sosemanuk to that version. For each cipher, two different implementations have been created. Dragon, Salsa20 and Sosemanuk are implemented in a function based version and a macro based version, respectively. The function based versions are realized with the goal of minimizing the use of flash memory. In contrast, the macro based versions are optimized to reach high throughput rates. LEX is treated as a special case.

Table 2: Specification of the most popular AVR devices (ATmega family)

Device	Flash [kbyte]	SRAM [byte]
ATmega8	8	1024
ATmega16	16	1024
ATmega32	32	2048
ATmega64	64	4096
ATmega128	128	4096
ATmega1281	128	8192

The version called 'LEX' is the transformation of the C version of LEX in Assembly language using five big static arrays. By contrast, the version named 'LEX V2' is an Assembly language implementation derived from the AES implementation [14].

3.1 Memory Usage

Table 3 shows the memory allocation in flash memory. In more detail, Table 3 provides (i) the size of the flash memory which is used to store the program code, (ii) the required size of flash memory for the storage of static arrays, like S-boxes for instance, (iii) the total size of program code and static arrays, and (iv) the associated AVR device, i.e., the smallest device on which the implementation of the cipher can be executed without errors. Table 4 focus on the SRAM consumption and provides the total size of required SRAM of each cipher, both in percentage and as absolute value. Considering the function based versions and LEX V2, we notice that flash memory needs are low for AES, Salsa20 and LEX, moderate for Dragon and high for Sosemanuk. High amounts of program code also typically indicate a high degree of implementation complexity. This is especially true for Sosemanuk. In terms of SRAM usage, AES, Salsa20, and LEX are again most efficient, followed by Dragon and Sosemanuk.

Table 3: Memory allocation in flash of Assembly implementations

Cipher	Program Code [byte]	Static Arrays [byte]	Total Memory [byte]	[percentage]	Device
AES	1154	266	1420	17,33%	ATmega8
Dragon (M)	25102	2048	27150	82,86%	ATmega32
Dragon (F)	4850	2048	6898	84,20%	ATmega8
LEX	1486	5120	6606	80,64%	ATmega8
LEX V2	1332	266	1598	19,51%	ATmega8
Salsa20 (M)	2984	0	2984	36,43%	ATmega8
Salsa20 (F)	1452	0	1452	17,72%	ATmega8
Sosemanuk (M)	44648	2048	46696	71,25%	ATmega64
Sosemanuk (F)	9092	2048	11140	67,99%	ATmega16

3.2 Performance

Performance benchmarks are provided in Table 5 and Table 6. Table 5 shows the number of cycles for the initialization, key setup, IV setup and encryption for each cipher. Table 6 focuses on the throughput of the encryption function for each cipher while Table 5 gives the number of cycles for one block size. The last column of Table 6 introduces a time-memory tradeoff metric, i.e., the product of the ratio of cycles per keystream byte (shown in Column 3 of Table 6) and the total amount of flash memory (shown in Column 4 of Table 3). Low values of this metric indicate high efficiency in the time-memory tradeoff. Considering the function based versions and LEX V2, high speeds are achieved by Sosemanuk, Dragon, LEX V2 while Salsa20 and AES

Table 4: Memory allocation in SRAM of Assembly implementations

Cipher	Total SRAM [byte]	[percentage]	Device
AES	224	21,88%	ATmega8
Dragon (M)	560	27,34%	ATmega32
Dragon (F)	560	54,69%	ATmega8
LEX	304	29,69%	ATmega8
LEX V2	304	29,69%	ATmega8
Salsa20 (M)	280	27,34%	ATmega8
Salsa20 (F)	280	27,34%	ATmega8
Sosemanuk (M)	712	17,38%	ATmega64
Sosemanuk (F)	712	69,53%	ATmega16

are significantly slower. In the time-memory tradeoff, LEX V2 is the most efficient followed by Salsa20 and AES.

Table 5: Performance of initialization, key setup, IV setup, encryption of Assembly implementations. All numbers given are measured CPU cycles

Cipher	Initialization	Key Setup	IV Setup	Encryption
AES	192	1535	57	5113
Dragon (M)	756	538	21232	16648
Dragon (F)	756	537	23680	17527
LEX	316	1484	5216	5502
LEX V2	313	1575	5595	5963
Salsa20 (M)	464	199	60	17812
Salsa20 (F)	460	199	60	18400
Sosemanuk (M)	514	14627	8559	8739
Sosemanuk (F)	519	15252	9143	9459

Table 6: Throughput of encryption of Assembly implementations

Cipher	Block Size [byte]	Ratio [cycles/byte]	Throughput [bytes/sec] @8MHz	Time Memory Tradeoff Metric [cycles/byte][byte]
AES	16	319,56	25034	453779
Dragon (M)	128	130,06	61509	3531197
Dragon (F)	128	136,93	58424	944541
LEX	40	137,55	58161	908655
LEX V2	40	149,08	53664	238222
Salsa20 (M)	64	278,31	28745	830485
Salsa20 (F)	64	287,50	27826	417450
Sosemanuk (M)	80	109,24	73235	5100954
Sosemanuk (F)	80	118,24	67660	1317166

4 Conclusion

This paper provides the first implementation results for Dragon, LEX, Salsa20 and Sosemanuk on 8-bit microcontrollers and therefore answers the question of how efficient candidates in the focus of eSTREAM Profile I (Phase 2) can be implemented on small embedded microcontrollers that are also constrained in memory resources. We confirm that all studied stream ciphers reach

higher speeds at keystream generation than AES. In terms of memory, Salsa20 and LEX can be implemented almost as compact as AES, while Dragon and Sosemanuk require noticeably more memory resources and may be sub-optimum for embedded applications with very low memory constraints. As an overall summary, considering the time-memory tradeoff metric, LEX and Salsa20 have turned out to yield significantly better results than AES.

References

[1] Call for Stream Cipher Primitives, Version 1.3, 12th April 2005. eSTREAM, ECRYPT Stream Cipher Project, 2005. Available from: `http://www.ecrypt.eu.org/stream/call/`.

[2] D. J. Bernstein. Notes on the ECRYPT Stream Cipher project (eSTREAM). Available from: `http://cr.yp.to/streamciphers.html`.

[3] eSTREAM – Update 1, September 2, 2005. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/057, 2005. Available from: `http://www.ecrypt.eu.org/stream/papersdir/057.pdf`.

[4] eSTREAM Optimized Code HOWTO. eSTREAM, ECRYPT Stream Cipher Project, 2005. Available from: `http://www.ecrypt.eu.org/stream/perf/`.

[5] FIPS 197. Announcing the AES, November 2001.

[6] D. J. Bernstein. Salsa20. Available from: `http://www.ecrypt.eu.org/stream/p2ciphers/salsa20/salsa20_p2.zip`.

[7] Alex Biryukov. A new 128-bit key stream cipher LEX. Available from: `http://www.ecrypt.eu.org/stream/p2ciphers/lex/lex_p2.zip`.

[8] Christophe De Cannire. eSTREAM testing framework. eSTREAM, ECRYPT Stream Cipher Project. Available from: `http://www.ecrypt.eu.org/stream/perf/`.

[9] K. Chen, M. Henricksen, W. Millan, J. Fuller, L. Simpson, E. Dawson, H. Lee, and S. Moon. Dragon: A fast word based stream cipher. Available from: `http://www.ecrypt.eu.org/stream/p2ciphers/dragon/dragon_p2.pdf`.

[10] Hongjun Wu. The stream cipher HC-128. Available from: `http://www.ecrypt.eu.org/stream/p2ciphers/hc256/hc128_p2.pdf`.

[11] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. SOSEMANUK, a fast software-oriented stream cipher. Available from: `http://www.ecrypt.eu.org/stream/p2ciphers/sosemanuk/sosemanuk_p2.pdf`.

[12] eSTREAM Short Report on the End of the Second Phase. eSTREAM, ECRYPT Stream Cipher Project, 2007.

[13] Brian Gladman. Brian Gladman's Home Page. Available from: `http://fp.gladman.plus.com/AES/index.htm`.

[14] Implementation of the AES Encryption on an 8-bit AVR microcontroller, Christian Roepke. Available from: `http://www.christianroepke.de/studium/praktikum_b/aes_encryption.asm`.

[15] Specification papers of the devices of the ATmega series. Available from: `http://www.atmel.com/dyn/products/devices.asp?family_id=607#760`.

[16] G. Meiser, T. Eisenbarth, K. Lemke-Rust, C. Paar. Software Implementation of eSTREAM Profile I Ciphers on embedded 8-bit AVR Microcontrollers (Paper at SASC 2007). Available from: `http://www.ecrypt.eu.org/stream/papersdir/2007/017.pdf`.

MPSoC-coupled Hardware Accelerator for Elliptic Curve Cryptography in Network Application Domain

Christoph Puttmann* and Jamshid Shokrollahi**

* Heinz Nixdorf Institute, University of Paderborn, Germany
puttmann@hni.upb.de

** Chair for System Security, University of Bochum, Germany
jamshid@crypto.ruhr-uni-bochum.de

Abstract

In this paper, we focus on a high performance hardware implementation for elliptic curve cryptography. Our hardware accelerator performs scalar multiplication in binary fields and is coupled to the on-chip network of a scalable multiprocessor system-on-chip (MPSoC), which is particularly suitable for network applications. Fast calculation is achieved by using a pipelined and parallelized Montgomery ladder algorithm based on Karatsuba finite field multiplication. Finally, implementation and performance results for different binary field sizes are presented for an ASIC and FPGA technology.

Keywords. Hardware Accelerator, MPSoC, Elliptic Curve Cryptography, Binary Fields

1 Introduction

In todays information age security becomes more and more important. Especially in the area of public key algorithms, Elliptic Curve Cryptography (ECC) gained increasing popularity. The reason for this is the high level of security with relatively small key sizes. However, due to the calculation complexity of ECC, application specific hardware implementations are required. This work focuses on the network application domain, where hardware accelerators for ECC are essential in order to ensure high data throughput. Therefore, we propose a hierarchical and scalable multiprocessor system-on-chip architecture, which comprises dedicated hardware accelerators for ECC that are coupled to an on-chip network. Moreover, the proposed structure can also efficiently be used in other application domains, *e.g.*, finding collisions on elliptic curves, or in the sieving phase of the number field sieve (NFS) method, when the elliptic curve operations are over rings of integers. Related work on high speed hardware implementations for ECC is published in [OP01], [GBS+03], and [AH06], with [AH06] reaching the highest performance. We show how faster polynomial multiplication using the Karatsuba method can improve the overall performance and integrate the developed hardware accelerator into a scalable multiprocessor system-on-chip.

The paper is structured as follows. Section 2 presents an overview of ECC over binary fields and the applied algorithms. In Section 3 the architectural concept of the proposed system is introduced. The implementation and performance results are shown in Section 4. Finally, Section 5 concludes the paper.

2 Elliptic Curves over Binary Fields

The general equation for a non-supersingular elliptic curve over the binary finite field $\mathbb{F}_{2^m}$ is given by equation:

$$y^2 + xy = x^3 + ax^2 + b \tag{1}$$

for appropriate parameters $a, b \in \mathbb{F}_{2^m}$. It is well known that the points $(x, y) \in \mathbb{F}_{2^m}^2$, which satisfy (1), together with the identity element $\mathcal{O}$, generate a group. The group operation is the addition of points, which can be geometrically represented by the tangent and chord operation. Let $\mathcal{P}_1 = (x_1, y_1)$ and $\mathcal{P}_2 = (x_2, y_2)$ be two given points on the curve in (1). Then the points $\mathcal{P}_3 = (x_3, y_3) = \mathcal{P}_1 + \mathcal{P}_2$ and $\mathcal{P}_4 = (x_4, y_4) = 2\mathcal{P}_1$ are computed using the following formulas:

$$\begin{array}{ll} \text{if } x_1 + x_2 \neq 0 & \text{if } x_1 \neq 0 \\ \lambda = \frac{y_1+y_2}{x_1+x_2} & \lambda = \frac{y_1}{x_1} + x_1 \\ x_3 = \lambda^2 + \lambda + x_1 + x_2 + a & x_4 = \lambda^2 + \lambda + a = x_1^2 + \frac{b}{x_1^2} \\ y_3 = (x_1 + x_3)\lambda + x_3 + y_1 & y_4 = (x_1 + x_4)\lambda + x_4 + y_1 \end{array}$$

When $x_1 + x_2 = 0$ or $x_1 = 0$ the results of addition and doubling, respectively, are $\mathcal{O}$. Using the above equations, we are able to multiply an integer k by a point $\mathcal{P}$, which is the result of $k - 1$ times adding the point to itself. This operation is known as scalar or point multiplication $k\mathcal{P}$. A very efficient method for scalar multiplication on elliptic curves is the Montgomery ladder algorithm. This method is shown in Algorithm 1 and was proposed for the first time in [Mon87] for factorization purposes and later applied to binary fields in [LD99].

Algorithm 1 The Montgomery point multiplication algorithm expressed in point level.

Input: An elliptic curve with a point $\mathcal{P}$ on it, together with the binary representation of the scalar multiplier k as $(k_{i-1}k_{i-2}\ldots k_1k_0)_2$.
Output: $k\mathcal{P}$

```
P1 ← P, P2 ← 2P
for j from i − 2 downto 0 do
    if k_j = 1 then
        P1 ← P1 + P2, P2 ← 2P2
    else
        P2 ← P1 + P2, P1 ← 2P1
    end if
end for
```

The equations above show the relations for addition and doubling of points when they are represented using the two coordinates x and y, which requires one inversion for any operation. Since inversion is an expensive task in finite fields, there are approaches, which avoid inversion by using projective coordinates. In projective coordinates points are represented as a triple (X, Y, Z) in such a way that $x \to X/Z^r$ and $y \to Y/Z^s$ for suitable parameters r and s. For any operation it is possible to choose Z in such a way that no inversion is required. Increasing the number of coordinates brings new multiplications, additions, and squarings into the computations. In efficient projective representations these extra number of operations are still fewer than the operations that are required for an inversion [HMV04].

The coordinates, which are processed during the calculation, are elements of a finite field $\mathbb{F}_{2^m}$. By optimal selection of the finite field representation as well as of the hardware structure that performs the operations, the efficiency of field arithmetic is also determined. For ECC the best representation is polynomial basis representation, in which arithmetic consists of multiplication of polynomials followed by a reduction modulo a sparse irreducible polynomial. The complexity of the latter task is much less than that of polynomial multiplication and hence it is desired to multiply polynomials as efficiently as possible. A very efficient method to multiply polynomials with degrees that are used in ECC is the Karatsuba algorithm, which was introduced for the first time by [KO63]. The three coefficients of the product $(a_1x + a_0)(b_1x + b_0) = a_1b_1x^2 + (a_1b_0 + a_0b_1)x + a_0b_0$ are "classically" computed with 4 multiplications and 1 addition from the four input coefficients a_1, a_0, b_1, and b_0. The Karatsuba formula uses only 3 multiplications and 4 additions:

$$(a_1x + a_0)(b_1x + b_0) = a_1b_1x^2 + ((a_1 + a_0)(b_1 + b_0) - a_1b_1 - a_0b_0)x + a_0b_0. \quad (2)$$

By applying (2) to larger polynomials the costs of extra additions vanish compared to those of the saved multiplications and an asymptotical cost of $O(m^{1.58})$ compared to the classical cost of $O(m^2)$ is achieved [HMV04].

3 Architectural Concept

The target platform for embedding our hardware accelerator is the so called GigaNetIC system [NP+07]. As depicted in Figure 1, the GigaNetIC architecture represents a hierarchical multiprocessor system-on-chip. At the lowest level of hierarchy, the *N-Core subsystem* comprises a $32\,bit$ RISC microprocessor with local memory. A parametrizable amount of these processor elements are connected via an on-chip wishbone bus. Together with a shared memory this forms a *cluster*, which represents the next higher level of hierarchy. At the top level of hierarchy, several clusters are connected via *switch boxes* to build an network-on-chip (NoC). This communication structure, the so called GigaNoC, features packet-switched wormhole routing and achieves up to $61\,GBit/s$ data throughput [PN+07]. The massively parallel structure of the GigaNetIC architecture is particularly suitable for network applications, because incoherent packet flows can be processed concurrently by the processor elements.

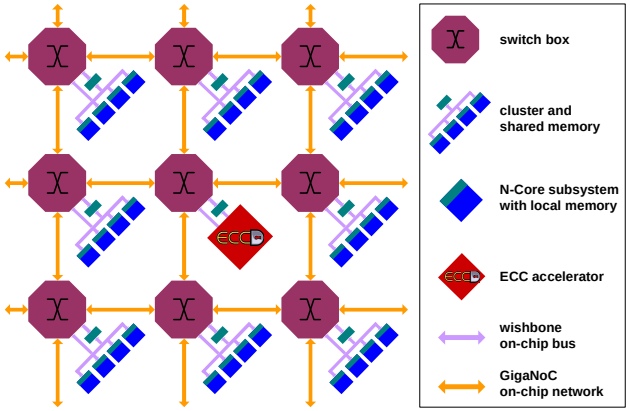


Figure 1: ECC accelerator coupled to the GigaNoC of the scalable GigaNetIC architecture.

In order to sustain the high data throughput, application specific hardware accelerator can be integrated at different hierarchy levels of the GigaNetIC system [NP+07]. To accelerate ECC we coupled a high performance implementation of the scalar multiplication to a switch box port at cluster level (cf. Fig. 1). In this way, every processor element can access the *ECC accelerator* via the GigaNoC.

The ECC accelerator calculates a scalar multiplication over the binary field $\mathbb{F}_{2^m}$. Moreover, the irreducible polynomial can easily be adapted, in order to support different field sizes. Throughout this work, we analyze the scalar multiplication for field sizes $m = 233$ and $m = 163$, which are specified in [Nat00]. We implemented a pipelined and parallelized Montgomery ladder

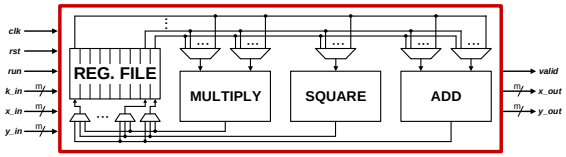


Figure 2: Core components of the ECC accelerator for calculation (*multiply, square, add*)

algorithm as described in [AH06]. This means, the point addition and point doubling (cf. Alg. 1) are executed almost in parallel. Furthermore, the finite field multiplier is implemented in pipelined fashion, so that no idle states occur and a minimum execution time is achieved. Instead of using a word serial multiplier, we applied the Karatsuba method for finite field multiplications. Thereby, the multiplication can be done by factor one third faster than in[AH06]. The structure of the ECC accelerator is depicted in Fig. 2. Although the scalar multiplication uses projective coordinates internally, the I/O interface supports affine coordinates. This means, the ECC accelerator features coordinate transformation including the final inversion step.

4 Implementation Results

The target technology for the GigaNetIC system comprising the proposed hardware accelerator is an ASIC implementation. Nevertheless, we also use FPGAs to implement prototypes and verify functional correctness. Table 1 shows the implementation results for a state of the art $65\,nm$ CMOS standard cell technology as well as for an Xilinx XC2V8000-4 FPGA target. Further details about the implementation of the GigaNetIC system itself can be found in [NP+07].

Table 1: Performance results of the hardware accelerator for an ASIC and FPGA implementation

	65nm CMOS @ 625 MHz		**XC2V8000-4 FPGA @ 50 MHz**	
Binary Field	**Size**	**Time**	**Size**	**Time**
233 bit (NIST)	$0.279\,mm^2$	$7.2\,\mu s$	$15,365$ slices	$90\,\mu s$
163 bit (NIST)	$0.222\,mm^2$	$5.1\,\mu s$	$10,477$ slices	$64\,\mu s$

The performance of the ECC accelerator is measured by means of execution time for a scalar multiplication including the coordinate transformation. The ASIC implementation achieves a clock frequency of $625\,MHz$. Depending on the size m of the binary field $\mathbb{F}_{2^m}$, a chip area of $0.279\,mm^2$ is required for $m = 233$ and $0.222\,mm^2$ for $m = 163$, respectively. Compared to [AH06] an absolute speedup of $21\mu s/5.1\mu s = 4.12$ is derived ($m = 163$), which still remains a speedup of 1.20 when normalizing the clock frequency. Since our design is not optimized for FPGA technology, *i.e.*, no Xilinix BlockRAM is used, we need roughly 2.4 times more slices than [AH06]. To calculate a scalar multiplication, our FPGA implementation is slower by a factor of 1.56, which is due to the clock frequency of $50\,MHz$. Again, when we normalize the clock frequency, we reach a speedup of 1.28 compared to [AH06].

5 Conclusion

In this paper, we have presented a high performance implementation of a scalar multiplication accelerator for elliptic curve cryptography by using the Karatsuba finite field multiplication within the Montgomery ladder algorithm for scalar multiplication. Our proposed ECC accelerator was implemented for different binary field sizes and mapped to an ASIC and FPGA technology. As a result, the scalar multiplication including coordinate transformation can be calculated in less than $10\mu s$, when realized in a state of the art $65\,nm$ CMOS technology.

Furthermore, the ECC accelerator was integrated in the GigaNetIC system, a hierarchical and scalable multiprocessor system-on-chip architecture that is particular suitable for network applications. Being coupled to the GigaNoC, the high performance network-on-chip of the GigaNetIC system, the elliptic curve scalar multiplier can be shared by several processors. In our future work, this can be used for collision finding on elliptic curves. Additionally, we will consider the proposed system for the sieving phase of NFS with elliptic curves over ring on integers.

Acknowledgement

Substantial parts of the research described in this paper was funded by the German Research Foundation (DFG) under project RU 477/8.

References

[AH06] Bijan Ansari and M. Anwar Hasan. High performance architecture of elliptic curve scalar multiplication. Tech. Report CACR 2006-01, Department of Electrical and Computer Engineering, University of Waterloo, Canada, 2006.

[GBS+03] C. Grabbe, M. Bednara, J. Shokrollahi, J. Teich, and J. von zur Gathen. A High Performance VLIW Processor for Finite Field Arithmetic. In Proc. of The 10th Reconfigurable Architectures Workshop (RAW-03), 2003.

[HMV04] Darrel Hankerson, Alfred J. Menezes, and Scott A. Vanstone. Guide to Elliptic Curve Cryptography. (Springer Professional Computing). Springer, Berlin, 2004.

[KO63] Anatoly A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. Soviet Physics Doklady, 7:595–596, 1963.

[LD99] Julio López and Ricardo Dahab. Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Precomputation. In CHES '99: Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems, pages 316–327, London, UK, 1999. Springer-Verlag.

[Mon87] Peter L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation, 48:243–264, 1987.

[Nat00] National Institute of Standards and Technology (NIST). Digital Signature Standard (DSS), volume FIPS 186-2, chapter Recommended elliptic curves for federal government use, pages 24–48. U.S. Department Of Commerce, 27 January 2000.

[NP+07] J.-C. Niemann, C. Puttmann, M. Porrmann, and U. Rückert. Resource Efficiency of the GigaNetIC Chip Multiprocessor Architecture. Journal of Systems Architecture (JSA), special issue on Architectural premises for pervasive computing, 53:285–299, May-June 2007.

[OP01] Gerardo Orlando and Christof Paar. A high-performance reconfigurable elliptic curve processor for $GF(2^m)$. Lecture Notes in Computer Science , 1965:41–56, 2001.

[PN+07] C. Puttmann, J.-C. Niemann, M. Porrmann, and U. Rückert. GigaNoC – A Hierarchical Network-on-Chip for Scalable Chip-Multiprocessors. In 10th Euromicro Conference on Digital System Design, Lübeck, Germany, 2007.

Finding message pairs conforming to simple SHA-256 characteristics: Work in Progress

Marko Hölbl*, Christian Rechberger**, Tatjana Welzer*

*Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia
http://www.feri.uni-mb.si
marko.holbl@uni-mb.si

**Institute of Applied Information Processing and Communications (IAIK), Graz University of Technology, Graz, Austria
http://www.iaik.tugraz.at/research/krypto

Abstract

Recent progress in hash functions analysis has led to collisions on the reduced version of SHA-256. Based on the analysis by Mendel *et al.*, we point out problems and challenges in automatic searching of message which fulfill the set of sufficient conditions of simple characteristics through (parts of) the SHA-256 compression function.

Keywords. SHA-256, sufficient conditions, collision search

1 Introduction

Owing to the recent cryptanalytic results on MD5 [WY05], SHA-1 [BCJCLJ05, RO05, WYY05] and similar hash functions, the influence of the attack on members of the SHA-2 family (*i.e.* SHA-224, SHA-256, SHA-384 and SHA-512) [FIPS02] is an important issue.

Although SHA-256 is considered as the successor of SHA-1 [FIPS02], it received little analysis in the cryptographic community [GH03, HPR04, MPPRR05, YB05]. In [MPRR06] Mendel *et al.* presented message pairs for a collision of SHA-256 reduced to 18 steps and a complex collision characteristic covering 19 steps. Their attacks is an extension of the analysis by Chabaud and Joux [CJ98] and Wang *et al.* [WYY05].

Using the set of sufficient conditions that describe local collisions in SHA-256, we develop a tool for automatic searching of such messages that conform to these characteristics. We came across several problems when developing the tool and present them in this paper.

2 Review of SHA-256

In the following we will use notation as described in Table 1. We only give a brief review of the structure of SHA-256. A complete description of SHA-256 can be found in [FIPS02].

SHA-256 is an iterated cryptographic hash function based on a compression function that updates the eight 32-bit state variables $A, \ldots, H$ according to the values of 16 32-bit words M_0, $\ldots$, M_{15} of the message. The compression function consists of 64 identical steps as presented in Fig. 1. The step transformation employs the bitwise Boolean functions f_{MAJ} and f_{IF}, and two GF(2)-linear functions

$$\Sigma_0(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)\,,$$
$$\Sigma_1(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)\,.$$

Table 1: Notation

notation	description
$A_i \ldots H_i$	state variables at step i of the compression function
$A \oplus B$	bit-wise XOR of state variable A and B
$A + B$	addition of state variable A and B modulo 2^{32}
$ROTR^n(A)$	bit-rotation of A by n positions to the right
$SHR^n(A)$	bit-shift of A by n positions to the right
N	number of steps of the compression function

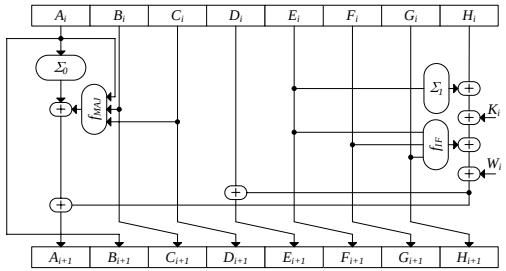


Figure 1: One step of the state update transformation of SHA-256.

The i-th step uses a fixed constant K_i and the i-th word W_i of the expanded message. The message expansion works as follows. An input message is padded and split into 512-bit message blocks. Let ME denote the message expansion function. ME takes as input a vector M with 16 coordinates and outputs a vector W with N coordinates. The coordinates W_i of the expanded vector are generated from the initial message M according to the following formula:

$$W_i = \begin{cases} M_i & \text{for } 0 \le i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & \text{for } 16 \le i < N \end{cases}$$

Taking a value for N different to 64 results in a step-reduced (or extended) variant of the hash function. The functions $\sigma_0(x)$ and $\sigma_1(x)$ are defined as follows: $\sigma_0(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$ and $\sigma_1(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$.

3 The message search and problems

According to Mendel *et al.* a 9-step local collision for SHA-256 is possible [MPRR06]. Furthermore, they give a characteristic (Table 2) from which a set of conditions can be derived. (Table 3, see also [DMPRR06]). Employing these conditions we develop a tool for automatic searching of messages which fulfill the presented sufficient conditions. During the design, we came across several problems.

Table 2: Example of a local collision for SHA-256 [MPRR06]

Step	W'	A'	B'	C'	D'	E'	F'	G'	H'	HW
01	80000000	80000000	0	0	0	80000000	0	0	0	2
02	22140240	0	80000000	0	0	20040200	80000000	0	0	5
03	42851098	0	0	80000000	0	0	20040200	80000000	0	5
04	0	0	0	0	80000000	0	0	20040200	80000000	5
05	80000000	0	0	0	0	80000000	0	0	20040200	4
06	22140240	0	0	0	0	0	80000000	0	0	1
07	0	0	0	0	0	0	0	80000000	0	1
08	0	0	0	0	0	0	0	0	80000000	1
09	80000000	0	0	0	0	0	0	0	0	0

Table 3: Sufficient conditions for the SHA-256 local collision given in Table 2. [DMPRR06]

no	type	condition
01	hard	$E_{04,09} = 0$
02	hard	$E_{04,18} = 0$
03	hard	$E_{04,29} = 0$
04	hard	$E_{04,31} = 1$
05	hard	$E_{05,09} = 1$
06	hard	$E_{05,18} = 1$
07	hard	$E_{05,29} = 1$
08	hard	$E_{07,31} = 0$
09	hard	$E_{08,31} = 1$
10	hard	$Z_{02,06} \oplus W_{02,06} = 1$
11	hard	$E_{03,09} \oplus W_{02,09} = 0$
12	hard	$Y_{02,09} \oplus W_{02,09} = 1$
13	hard	$E_{03,18} \oplus W_{02,18} = 0$
14	hard	$Y_{02,18} \oplus W_{02,18} = 1$
15	hard	$Z_{02,20} \oplus W_{02,20} = 1$
16	hard	$Z_{02,25} \oplus W_{02,25} = 1$
17	hard	$E_{03,29} \oplus W_{02,29} = 0$
18	hard	$Y_{02,29} \oplus W_{02,29} = 1$
19	hard	$A_{01,31} \oplus A_{00,31} = 0$
20	hard	$E_{01,31} \oplus E_{00,31} = 0$
21	hard	$Z_{03,03} \oplus W_{03,03} = 1$
22	hard	$Z_{03,04} \oplus W_{03,04} = 1$
23	hard	$Z_{03,07} \oplus W_{03,07} = 1$
24	hard	$E_{02,09} \oplus E_{01,09} = 0$
25	hard	$Z_{03,12} \oplus W_{03,12} = 1$
26	hard	$Z_{03,16} \oplus W_{03,16} = 1$
27	hard	$E_{02,18} \oplus E_{01,18} = 0$
28	hard	$Z_{03,18} \oplus W_{03,18} = 1$
29	hard	$Z_{03,23} \oplus W_{03,23} = 1$
30	hard	$Z_{03,25} \oplus W_{03,25} = 1$
31	hard	$E_{02,29} \oplus E_{01,29} = 0$
32	hard	$A_{03,31} \oplus A_{01,31} = 0$
33	hard	$A_{04,31} \oplus A_{03,31} = 0$
34	hard	$Z_{06,06} \oplus W_{06,06} = 1$
35	hard	$Z_{06,20} \oplus W_{06,20} = 1$
36	hard	$Z_{06,25} \oplus W_{06,25} = 1$
37	hard	$E_{05,31} \oplus E_{04,31} = 0$
38	hard	$E_{03,31} \oplus Z_{03,30} \oplus W_{03,30} = 1$
39	easy	$W_{02,09} \oplus W_{06,09} = 1$
40	easy	$W_{02,18} \oplus W_{06,18} = 1$
41	easy	$W_{02,29} \oplus W_{06,29} = 1$

In Table 3 Z and Y denote the outputs of $\sigma_0(x)$ and $\sigma_1(x)$. When talking about bit overlaps of sufficient conditions, we have consider this fact.

Firstly, SHA-256 employs a complex step update operation (Fig. 1). It simultaneously updates two state variables, thus makes message search difficult. Hence, we have to consider that message word W_i influences both state variables A_i and E_i.

Secondly, particular sufficient conditions interfere with each other. A specific bit of a state variable is included in multiple sufficient conditions in the same step. An example two such conditions are sufficient conditions no. 12 ($Y_{02,09} \oplus W_{02,09} = 1$) and no. 14 ($Y_{02,18} \oplus W_{02,18} = 1$). This limits the capabilities for bit changing through message word modification. We solve the problem by searching for same bits in sufficient conditions of the same step. If a part of the current sufficient condition includes the same bit for the same step as previous ones, we skip it. An exception are bit overlaps of message words W_i and state variables A_i resp. E_i. In this case, the specific bits can be changed.

Thirdly, some sufficient conditions include not only state variable conditions but also message word conditions which influence sufficient conditions in successive steps. Sufficient condition no. 10 is such an example ($Z_{02,06} \oplus W_{02,06} = 1$) When making message modification of W_i in successive steps we also have to consider these. In case we change the message word W_i (step i), we would have to check the sufficient conditions in the previous step (namely $i - 1$). Thus, we have limited message modification possibilities.

Fourthly, there are special types of sufficient conditions, *e.g.* no. 11 ($E_{03,09} \oplus W_{02,09} = 0$). This type of conditions include a state variable from the current (*e.g.* A_i resp. E_i) step and a message word $W_i - 1$ from the previous step. In this case it is impossible to modify message word from step $i - 1$ in order to flip a bit in step i. In such cases we have to generate a new random message word and try again from the beginning.

In some special cases bit carries play an important role. Nevertheless bit i is changed in W_i, which should influence the same bit in the state variable of step $i + 1$, the flip also influences other bits. In such cases no special technique is employed.

4 Conclusion and outlook

We presented the problems of finding message pairs conforming to simple characteristics for SHA-256. Several problems arise, which were discussed. We identified four main classes of problems, namely the complex structure of the step operation of SHA-256, interference between sufficient conditions, special cases of sufficient conditions, interference between message word and state variables in sufficient conditions and carry bits influence. That is why often random generation of message words must be employed to pass the specific sufficient conditions. Future work includes optimizing the algorithm for more efficient search of messages.

References

[BCJCLJ05] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 36–57. Springer, 2005.

[CJ98] Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, SA, August 23-27, 1998, Proceedings*, volume 1462, pages 56–71. Springer, 1998.

[DMPRR06] Christophe De Cannière, Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. *SHA Evaluation Report for CRYPTREC*, 2006/01/21.

[GH03] Henri Gilbert and Helena Handschuh. Security analysis of SHA-256 and sisters. In Mitsuru Matsui and Robert Zuccherato, editors, *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003, Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*, pages 175–193. Springer, 2003.

[HPR04] Philip Hawkes, Michael Paddon, and Gregory G. Rose. On corrective patterns for the SHA-2 family. Cryptology ePrint Archive, Report 2004/207, August 2004. `http://eprint.iacr.org/`.

[MPPRR05] Krystian Matusiewicz, Josef Pieprzyk, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of simplified variants of SHA-256. In *Proceedings of WEWoRC 2005*, LNI P-74, pages 123–134, 2005.

[MPRR06] Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of Step-Reduced SHA-256. In Vincent Rijmen, editor, *Fast Software Encryption - FSE 2006, 13th International Workshop, Graz, Austria, March 15-17, 2006, Proceedings*, volume 4047 of *LNCS*, pages 126–143, Springer, 2006.

[FIPS02] National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002. Available online at `http://www.itl.nist.gov/fipspubs/`.

[RO05] Vincent Rijmen and Elisabeth Oswald. Update on SHA-1. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *LNCS*, pages 58–71. Springer, 2005.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.

[WY05] Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

[YB05] Hirotaka Yoshida and Alex Biryukov. Analysis of a SHA-256 variant. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography (SAC 2005), Kingston, Ontario, Canada, August 11-12, 2005, Proceedings to appear*, volume 3897 of *Lecture Notes in Computer Science*, pages 245–260. Springer, 2005.

XOR-Hash : A Hash Function based on XOR

Stéphane Manuel and Nicolas Sendrier

Projet Codes, INRIA Rocquencourt
BP 105, 78153 Le Chesnay - Cedex, France
Stephane.Manuel@inria.fr, Nicolas.Sendrier@inria.fr

Abstract

We propose here a new iterative hash function design. Its compression function is based on a one-way function $f : \{0,1\}^r \to \{0,1\}^r$ which is applied on the chaining variable and on a fixed number $t-1$ of message blocks. The t values are then xored to form the next chaining variable. This compression function is iterated using the Merkle-Damgard construction and a final output transformation g reduces the hash value to a size of $n = 2r/(2+\log_2 t)$ bits. Under proper assumptions on f and g we claim a security of $r/(2+\log_2 t) = n/2$ bits against collision search.

1 Introduction

Recent works on hash function cryptanalysis [BCJ+05, WY05, DR06] have stressed the importance of designing new efficient and secure cryptographic hash functions.

Most currently used hash functions are based on compression functions. Damgard and Merkle [Dam89, Mer89] have provided a domain extender which permits to iteratively construct a hash function from a given compression function. It was shown that the resulting hash function is at least as resistant to collision attacks as the underlying compression function.

There are two main approach for designing compression functions. Either dedicated designs, like the MD4-family, or based on block ciphers [BRS02, PGV93, PGMR06].

We propose a new compression function design based on a single one-way function $f : \{0,1\}^r \to \{0,1\}^r$. The function f is applied on the chaining variable and on a fixed number $t-1$ of message blocks. The t values are then xored to form the next chaining variable. We use the Merkle-Damgard domain extender with a final output transformation g. By limiting the number t of blocks to be xored, we relate the security of our scheme to difficult coding theory problems, namely decoding in a random linear code. For the parameter we consider the best decoding attack is Wagner's generalized birthday attack [Wag02]. This will give us a security of $r/(2+\log_2 t)$ bits against collision search. In order to get a n bit hash function with $n/2$ bits of security, we will take the final output transformation with r bits of input and $n = 2r/(2+\log_2 t)$ bits of output.

The paper is organized as follows. In section 2 we introduce related works. We present our construction in section 3 and we give a security analysis in section 4. We then draw our conclusions.

2 Related Works

In this section, we will introduce two related works which conducted us to create our design: XOR-MAC and the Fast Syndrome Based hash functions. In 1995, Bellare *et al.* [BGR95] proposed new methods for message authentication codes called XOR schemes. This schemes use as underlying primitive a finite pseudorandom functions (PRF) which can be defined from a block cipher or from the compression function of a cryptographic hash function. The computation of a XOR-MAC consists of three steps: (1) encode the message as a collection of blocks; (2) apply

the finite PRF to each of the blocks to obtain a collection of PRF images; and (3) XOR the set of PRF images together, building the MAC out of the result. Bellare *et al.* proved that for a secure finite PRF family, the XOR-MACs schemes based on it are also secure. However, the security model of message authentication codes is different from the one of hash functions. As itself, XOR-MAC can not be directly converted in a secure hash function.

In 2005, Augot *et al.* [AFS05] proposed a family of iterative hash functions based on codes. The compression function is a syndrome mapping, which consists in the XOR of a small number t of columns of a large binary matrix given as parameter. The collision resistance of this compression function can be reduced to difficult coding problems (syndrome decoding). The compression function is scalable and its parameters are determined with respect to the Wagner's generalized birthday attack. This attack is the best known attack on this scheme.

In our construction, we will use the simplicity of the XOR-MAC design in order to achieve efficiency, and we will give a security analysis related to the one of the Fast Syndrome Based hash function in order to achieve security.

3 The proposed hash function

3.1 Preliminaries and notations

Let n, r and t be positive integers. Our main building block will be a (one-way) mapping

$$f : \{0,1\}^r \to \{0,1\}^r.$$

We will also use a final compression function $g : \{0,1\}^r \to \{0,1\}^n$ (thus $n < r$). We will discuss the security requirement of f and g in §4. The parameter t will be the number of blocks to be added in the compression function.

- For all integer $i \in \{0, \ldots, t-1\}$ we denote by $\langle i \rangle$ the natural binary encoding of i as a $\lceil \log_2 t \rceil$-bit string.
- We will denote by $x \parallel y$ the concatenation of the bit string x with the bit string y.
- For a bit string x of length r, $\tilde{x}$ denotes the $r - \lceil \log_2 t \rceil$ rightmost bits of x.

3.2 Construction

The principle of our construction is to build a secure and efficient cryptographic hash function based on a one-way function f from r bits to r bits and the XOR of the images by f of blocks of the message. As in XOR-MAC, we prefix each block by $\langle i \rangle$, $0 \leq i < t$, in order to prevent an attack by block permutation.

Let m be the message to be hashed and assume that it includes a prefix-free padding. Let $m = m_1 m_2 \ldots m_l$, where each m_i has length $r - \lceil \log_2 t \rceil$. The initial value IV has length $r - \lceil \log_2 t \rceil$. Details of the scheme are given in Figure 1. Note that we do not instantiate functions f and g. We will just make a few security assumptions on them in the next section.

4 Security analysis

In this section, we will consider the security of our construction. For an ideal cryptographic hash function h outputting n bits, a preimage or 2nd-preimage attack should require $\mathcal{O}(2^n)$ evaluations of the hash function, and a collision attack should require $\mathcal{O}(2^{n/2})$ evaluations of the hash function. According to the Handbook of applied cryptography [MOV96], a secure hash function requires that preimage, 2nd-preimage and collision attacks are computationally infeasible. The security objective of our construction is to build a secure hash function. The


```
Procedure XOR-hash
input: IV, m_1, ..., m_l ∈ {0,1}^{r-⌈log_2 t⌉}
    H ← f(⟨0⟩ ‖ IV)
    i ← 1 ; j ← 1
    while i ≤ l
        if j = t
            H ← f(⟨0⟩ ‖ H̃)
            j ← 1
        H ← H ⊕ f(⟨j⟩ ‖ m_i)
        i ← i + 1 ; j ← j + 1
    return g(H)
```

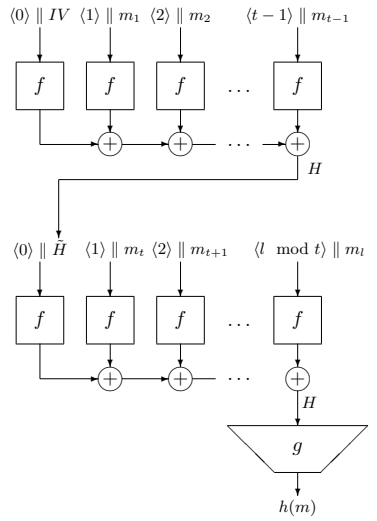


Figure 1: The XOR-hash function

cost of preimage, 2nd-preimage and collision should be beyond the computational power of an adversary.

Problem 1 **(t-sum preimage for f)** Given $s \in \{0,1\}^r$, find $x_1, \ldots, x_l$ distinct in $\{0,1\}^r$ with $0 < l \leq t$ such that $f(x_1) \oplus \ldots \oplus f(x_l) = s$.

Problem 2 **(t-sum collision for f)** Find $x_1, \ldots, x_l$ distinct in $\{0,1\}^r$ with $0 < l \leq 2t$ such that $f(x_1) \oplus \ldots \oplus f(x_l) = 0$.

Problem 3 **(t-sum 2nd preimage for f)** Given $x_1, \ldots, x_l$ in $\{0,1\}^r$ with $l \leq t$, find $x'_1, \ldots, x'_{l'}$ distinct in $\{0,1\}^r$ with $0 < l' \leq t$ such that $f(x_1) \oplus \ldots \oplus f(x_l) = f(x'_1) \oplus \ldots \oplus f(x'_{l'})$.

Problem 4 **(preimage for g)** Given $y \in \{0,1\}^n$, find $x \in \{0,1\}^r$ such that $g(x) = y$.

Problem 5 Find s in $\{0,1\}^r$ and $x_1, \ldots, x_l$ distinct in $\{0,1\}^r$ with $0 < l \leq t$ such that $f(x_1) \oplus \ldots \oplus f(x_l) \neq s$ and $g(f(x_1) \oplus \ldots \oplus f(x_l)) = g(s)$.

Problem 6 Given s in $\{0,1\}^r$, find $x_1, \ldots, x_l$ distinct in $\{0,1\}^r$ with $0 < l \leq t$ such that $f(x_1) \oplus \ldots \oplus f(x_l) \neq s$ and $g(f(x_1) \oplus \ldots \oplus f(x_l)) = g(s)$.

Problems 5 and 6 involve both f and g. In some sense, they express the "independence" of f and g. Resistance to Problem 5 is some kind of collision resistance for g relatively to f. Resistance to Problem 6 is some kind of 2nd preimage resistance for g relatively to f.

Sketch of the security arguments:

1. We claim that preimage, 2nd preimage and collision resistance for our hash function proposal can be reduced to some of the six above problems.

2. We assume the best attack on Problems 1 to 3 is Wagner's generalized birthday attack which has complexity at least $O\left(2^{r/(1+\log_2 t)}\right)$ for Problems 1 and 3, and complexity at least $O\left(2^{r/(2+\log_2 t)}\right)$ for Problem 2.

3. We assume that g is such that Problems 4 to 6 are harder than Problems 1 to 3.

Under those claims/assumptions, our scheme provides at least $r/(2+\log_2 t)$ bits of security against collision search. We will thus choose n (the output size of g) to be equal to $2r/(2+\log_2 t)$.

We give some instance of the parameters in Table 1.

Table 1: XOR-hash security parameters against collision search

security	r	t	n
2^{80}	240	2	160
2^{80}	320	4	160
2^{128}	384	2	256
2^{128}	512	4	256
2^{128}	640	8	256
2^k	$k \times (2+\ell)$	2^ℓ	$2k$

5 Conclusions

We have presented a new hash function construction. Its security is related to the difficulty, for a given one-way function $f : \{0,1\}^r \to \{0,1\}^r$, of finding a set of values $x_1, \ldots, x_l$ such that $f(x_1) \oplus \cdots \oplus f(x_l)$ has a prescribed value and $l \leq 2t$ for some fixed parameter t. This relates to difficult coding theory problems, and the present state of the art indicates that, if f is a pseudo-randon function, the best solving method is Wagner's generalized birthday attack.

This leads to large values of r. Finding a secure function f handling large blocks might be difficult and/or unefficient. However, our design do not require a "full" security of r bits.

References

[AFS05] D. Augot, M. Finiasz and N. Sendrier, A Family of Fast Syndrome Based Cryptographic Hash Functions. *International Conference on Cryptology in Malaysia - Mycrypt 2005*, *Lecture Notes in Computer Science 3715*, pages 64-83. Springer Verlag, 2005.

[BGR95] M. Bellare, R. Guerin and P. Rogaway, XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. *Advances in Cryptology - Crypto 95*, *Lecture Notes in Computer Science 963*, pages 15-20. Springer Verlag, 1995.

[BRS02] J. Black, P. Rogaway and T. Shrimpton, Black-Box Analysis of the Block-Cipher-Based Hash Function Constructions from PGV. *Advances in Cryptology - Crypto 02*, *Lecture Notes in Computer Science 2442*, pages 320-335. Springer Verlag, 2002.

[Dam89] I. Damgard, A Design Principle for Hash Functions. *Advances in Cryptology - Crypto 89*, *Lecture Notes in Computer Science 435*, pages 416-427. Springer Verlag, 1989.

[DR06] C. De Caniere and C. Rechberger, Finding SHA-1 Characteristics: General Results and Applications. *Advances in Cryptology - Asiacrypt 06*, *Lecture Notes in Computer Science 4284*, pages 1-20. Springer Verlag, 2006.

[BCJ+05] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet and W. Jalby, Collisions of SHA-0 and Reduced SHA-1. *Advances in Cryptology - Eurocrypt 05*, *Lecture Notes in Computer Science 3494*, pages 36-57. Springer Verlag, 2005.

[MOV96] A. Menezes, P.C. van Oorschot and S.A. Vanstone, Handbook of Applied Cryptography. CRC Press, 1996.

[Mer89] R.C. Merkle, One Way Hash Function and DES. *Advances in Cryptology - Crypto 89*, *Lecture Notes in Computer Science 435*, pages 428-446. Springer Verlag, 1989.

[PGMR06] T. Peyrin, H. Gilbert, F. Muller and M. Robshow, Combining Compression Functions and Block Cipher-Based Hash Functions. *Advances in Cryptology - Asiacrypt 06*, *Lecture Notes in Computer Science 4284*, pages 315-331. Springer Verlag, 2006.

[PGV93] B. Preneel, A. Bosselaers and J. Vandewalle, Hash Functions Based on Block Ciphers: A Synthetic Approach. *Advances in Cryptology - Crypto 93*, *Lecture Notes in Computer Science 773*, pages 368-378. Springer Verlag, 1993.

[Wag02] D. Wagner A Generalized Birthday Problem. *Advances in Cryptology - Crypto 02*, *Lecture Notes in Computer Science 2442*, pages 288-303. Springer Verlag, 2002.

[WY05] X.Y. Wang and H.B. Yu, How to break MD5 and other hash functions. *Advances in Cryptology - Eurocrypt 05*, *Lecture Notes in Computer Science 3494*, pages 19-35. Springer Verlag, 2005.

Efficient Hash Collision Search Strategies on Special-Purpose Hardware

Tim Güneysu, Christof Paar, Sven Schäge

Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
{gueneysu,cpaar}@crypto.rub.de, sven.schaege@nds.rub.de

Abstract

Hash functions play an important role in various cryptographic applications. Modern cryptography relies on a few but supposedly well analyzed hash functions, most of which are part of the so called MD4-family. This work shows whether it is possible, using special-purpose hardware, to significantly speedup collision search for MD4-family hash functions. A thorough analysis of the computational requirements of MD4-family hash functions and their corresponding collision attacks reveals that a microprocessor based architecture is best suited for the implementation of collision search algorithms. Consequently, we designed and implemented a general-purpose microprocessor for MD4-family hash-functions with minimal area requirements and, based on this, a full collision search integrated circuit. Comparing the performance characteristics of both ASICs with standard PC processors and networks, it turns out that our design, massively parallelized, is nearly four times more cost-efficient than parallelized standard PCs. We believe that there is much room for further improvements left.

Keywords. Hash functions, special-purpose hardware, crypto attacks.

1 Introduction

In February 2005 Wang et al. presented a new attack method against the popular Secure Hash Algorithm (SHA-1). It reduces the computational attack complexity to find a collision from $O(2^{80})$ to approximately $O(2^{69})$ [10] leading to the announcement that SHA-1 has been broken in theory. Soon it was further improved to $O(2^{63})$ [9]. However, still this attack is supposed to be theoretic in nature, because the necessary number of computations is very high.
For practical attacks, all theoretical results have to be mapped to an executable algorithm, which subsequently has to be launched on an appropriate architecture.
Basically, there are two ways to design such architectures, namely standard and special-purpose hardware. Generally, both FPGA and ASIC architectures require higher development costs than PC based systems. However, when manufactured at high volumes they quickly excel PC clusters with respect to cost-efficiency.
The main issue of this thesis is whether it is possible to develop alternative hardware architectures for collision search which offer better efficiency than standard PC architectures. Given a certain amount of money, which hardware architecture should be invested in to gain best performance results for collision search? Our solution is a highly specialized, full collision search unit called μCS which is based on a very compact ASIC microprocessor referred to as μMD.
In the cryptographic literature, there are few works that deal with practical aspects of collision search for MD5-family hash functions. Most of the contributions on collision search are dedicated to rather theoretical problems. To our best knowledge, this is the first work that analyzes implementation requirements for collision search algorithms from a computational perspective.
This work is structured as follows. Section 2 gives an introduction to the basic features of MD4-family hash functions and their attacks. In contrast to this, Section 3 presents the design details of μMD and μCS. In Section 4, we develop a metric for adequately describing the cost-efficiency of hardware units with respect to collision search. We then use this metric to compare our solution with a standard Pentium 4 based PC system. We close with a short conclusion.

2 Attacks on Hash Functions of the MD4-family

A (cryptographic) hash function is an efficiently evaluable mapping h which maps arbitrary-sized messages to fixed-size hash values [2]. There are at least three features a secure hash function is expected to have; (first) preimage resistance, second preimage resistance, and collision resistance. Successful attacks on collision resistance, i.e. finding two distinct messages that map to the same hash value, are much more promising, and so most attacks in the literature focus hereon.
As it is hard to efficiently describe algorithms that directly process inputs of variable length, hash functions of the MD4-family first divide the input message M into fixed-size message blocks, which are successively processed by a so called compression function. To impose chaining dependencies between consecutive message blocks, the compression function also processes the output of the immediately preceding computation, which is in this context also called chaining value. The first chaining value is a fixed initialization vector IV. The output of the computation of the last message block is defined to be the output of the entire hash function. Hash functions of the MD4-family mainly differ in their initialization vectors and their compression functions.
Generally, there are two types of attacks on MD4-family hash functions, generic and specific attacks. Generic attacks are attacks that are applicable to all (even ideal) hash functions. Specific attacks try to exploit the knowledge of the inner structure of the hash function and its inherent weaknesses. In this way, it is possible to *construct* collisions to a certain extent. Specific attacks are always dedicated to a single concrete hash function. However, there are some general design principles for developing specific attacks for MD4-family hash functions. Such attacks can be divided into two phases. The first phase launches a differential attack [1] on the inner structure of the compression function. The second phase consists of usefully utilizing the remaining freedom of choice for the concrete message bits. This freedom can of course be used to predefine parts of the input messages. However, another and very popular application is to exploit it for a *significant* acceleration of the collision search. These techniques are called single step modifications, multi step modifications, and tunneling [4, 5, 11].
For MD5 [7], there exist several efficient collision search algorithms [3, 4, 8] constructed according to these principles. Joščák [3] compares their performances in more detail, while Klima's collision search (CS) approach [4] turns out to be the fastest. In contrast to the other ones, this algorithm extensively makes use of tunneling. As an example and for comparison reasons, we fully implemented CS into our μCS unit.

3 Collision Search Processor Design

When analyzing MD4-family hash functions and their corresponding collision search algorithm, one can find several important hints on how a suitable hardware architecture should look like. First, MD4-family hash functions have been developed for 32-bit processor systems. Our target architecture should also support 32-bit operands, otherwise expensive correction operations have to be applied. Second, collision search algorithms can hardly be parallelized on lower hierarchical levels, since most operations also compute the result of their predecessor. Third, tunneling will, besides multi step modifications, become a standard means for improving collision search based on differential patterns. Unfortunately, tunneling highly parameterizes the computation path using loop constructions. This requires efficient resource reuse, while at the same time making usual hardware acceleration techniques like pipelining hardly useful. We finally decided to implement the collision search algorithm CS on a 32-bit microprocessor based ASIC architecture. The central microprocessor we call μMD, the final collision search circuit μCS. μMD uses a very small instruction set, consisting of not more than sixteen native commands. As we aimed at a entirely compact solution, we maximized reuse of program code wherever possible. As a result, we also implemented a sufficiently large hardware stack and indirect load and store operations.

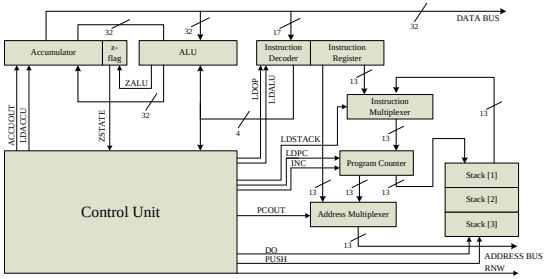


Figure 1: Inner structure of processor

In combination, they provide for a comfortable access to parameterized subfunctions. μMD is based on the design approach from [6]. The instruction set is adapted according to the requirements of the target application. Therefore, several instructions have been deleted, others have been added.
μCS is our final integrated circuit for collision search. Roughly spoken, it consists of a single μMD unit, additional memory and I/O logic. To start a computation, it just needs an initial seed for its integrated PRNG. When a collision is found, the corresponding message words are returned. Except for the PRNG initialization phase and the collision output sequence, there is no further I/O communication required. This decisively supports parallelization approaches making the overhead for additional control logic negligible.

4 Implementation

Implemented on an FPGA, μMD is comparably small. It uses about 9 percent of the slices of a Spartan3 xc3s1000 FPGA. The final clock frequency is about 95 MHz after synthesis. Synthesizing μCS for standard cells (UMC 130 nm) requires 0.959618 mm^2 chip area. Contrarily, μMD can be realized with only 0.026626 mm^2. This is just 2.77% of the chip area for the full μCS unit. μMD requires about 6000 gate equivalents (GE), μCS circa 210000 GE. For comparison, we also tested collision search on a standard PC processor. We used a Pentium 4, 2.0 GHz machine (Northwood core) with approximately 55 million transistors build in 130 nm circuit technology. Its die size is 146 mm^2.
We define the time $T = \frac{C}{f}$ to be the average time for a single unit to find a collision. Unfortunately, μMD and μCS are not available for a practical test run. For μCS, T is computed based on the average number C of cycles required to find a collision and the frequency f.
After implementing an assembler version of CS, we estimate CS executed on μMD to require roughly eight times more clock cycles than on a Pentium 4. Assuming equal production constraints, meaning the same price per chip area, each of our solution is much more effective than a comparable Pentium 4 architecture. The higher speed of a Pentium 4 processor is achieved only with much more area. The area-time product P reflects this fact appropriately.
ting an assembler version of CS, we estimate CS executed on μMD to require roughly eight times more clock cycles than on a Pentium 4. Assuming equal production constraints, meaning the same price per chip area, each of our solution is much more effective than a comparable Pentium 4 architecture. The higher speed of a Pentium 4 processor is achieved only with much more area. The area-time product P reflects this fact appropriately.

	Pentium 4	μCS	μMD
Q_A	$0.3425\frac{€}{\text{mm}^2}$	$0.3425\frac{€}{\text{mm}^2}$	$0.3425\frac{€}{\text{mm}^2}$
C	$60{\cdot}10^9$	$480{\cdot}10^9$	$480{\cdot}10^9$
f	2GHz	102.9 MHz	228.8 MHz
T	30 s	4660.8 s	2097.6 s
A	146 mm^2	0.959618 mm^2	0.026626 mm^2
P	4380	4472.6	55.9
Q_s	50 €	0.3287 €	-
O_p	300 %	5 %	-
Q_p	200 €	0.3451 €	-
R	6000	1608.4	-

Table 1: Cost overview

It is also interesting to compare the Pentium 4 with our *full* collision search unit μCS. One can see that a single unit of μCS is just 1.021 times less efficient than a Pentium 4 processor, although it includes all logic and data required to immediately start a collision search.
When we use an off-the-shelf standard PC for parallelization, we have to consider the costs for I/O, ROMs, motherboards, fans, and additional equipment for parallelization like network cards and cables. Altogether, we believe the costs to be at least 200 €, whereas we assume the Pentium 4 2.0 GHz to have a price of roughly 50 €. As a consequence, we estimate the price per chip Q_A area to be $Q_A = \frac{50\ €}{146\ \text{mm}^2} = 0.3425\frac{€}{\text{mm}^2}$. Using such off-the-shelf standard PCs, the processors are connected to each other by standard network equipment. The costs per unit will increase four-fold. This means, that for each single parallelized Pentium 4 processor one can buy four boxed ones. In other words, the parallelization overhead O_p is 300 %.
The parallelization of μCS requires only few additional logic per unit. For our solution, we assume the overhead in area to almost negligible, i.e. ranging below 5 %.
Based on these considerations, we believe that our *full* collision search solution is noticeably more effective for collision search than parallelized Pentium 4 processors. Our estimates are summed up in Table 1. Q_s reflects the price for a single standalone unit of the corresponding architecture. In contrast, Q_p is the average price for a single unit after parallelization. $R = Q_p{\cdot}T$ then reflect the amount of money required to find a collision within one second.
Obviously, for finding a single MD5 collision per second one has to spend 6000 € in (30) parallelized standard PCs. Assuming similar constraints for manufacturing ASICs, the price for the same performance invested in parallelized μCS units is 1608.4 €. So, it is almost four times more cost-efficient than the Pentium 4 standard PC architecture.

4.1 Estimations for SHA-1

We believe that a comparable implementation of a SHA-1 collision search algorithm in dedicated and parallelized collision search units has similar performance characteristics.
The average number of required clock cycles to find a collision is primarily dependent on the available theoretical results. Currently, attacks on SHA-1 have a complexity of about $O(2^{63})$. For practical attacks, we believe this number to be still very large.
Given 1 mio. €, an attacker can buy enough standard PC equipment to on average find a MD5 collision within $\frac{6000}{1000000} = 0.006$ s. Invested in our collision search units, it would take only $\frac{1608.4}{1000000} = 0.0016084$ s. Finding MD5 collisions based on CS has in [3] reported to be have a complexity of about $7.7 \cdot 2^{30}$ MD5 step operations. The current bound for SHA-1 collisions is $O(2^{63})$. Assuming a complexity of exactly 2^{63} step operations and a similar execution time for a

single step operation, finding collisions for SHA-1 takes about 2^{30} times more time than for MD5. We can thus conclude how long it would take to find a single SHA-1 collision with equipment for 1 mio. €. Invested in standard PCs, it would take $0.006 \text{ s} \cdot 2^{30} = 6442450$ s or almost 75 days. Using our collision search units, this time would be only $0.0016084 \text{ s} \cdot 2^{30} = 1727006$ s or roughly 20 days.

5 Conclusion

In this work we analyzed the hardware requirements of current and future collision search algorithms for hash functions of the MD4-family. We used our results to develop an appropriate hardware platform which executes collision search algorithms about four times faster than standard PC architectures.
The heart of our design is a very small microprocessor μMD with only sixteen instructions. At the same time, it provides very effective means to support program code reuse, what greatly helps to keep the size of our overall collision search unit μCS small.
In the context of MD4-family hash functions, μMD is general-purpose, meaning that it is appropriate for the execution of all MD4-family hash functions and also of all corresponding current and future collision search algorithms.
In contrast to standard PCs, the final collision search unit needs only very little additional logic. This reduces its price and greatly eases parallelization approaches.
We believe that our design approach is much better suited for collision search than standard PCs. When money is spent on collision search, our design, massively parallelized, is nearly four times more cost-efficient than parallelized P4 standard PCs. Given 1 million €, our solution can so find a SHA-1 collision in about 20 days, whereas a Pentium 4 based architecture would take nearly 75 days.

References

[1] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*, 1993.

[2] M. Daum. *Cryptanalysis of Hash Functions of the MD4-Family.* PhD thesis, Ruhr-Universität Bochum, 2005.

[3] D. Joščák. Finding collisions in cryptographic hash functions. Master's thesis, Univerzita Karlova v Praze, 2006.

[4] V. Klima. *Tunnels in Hash Functions: MD5 Collisions Within a Minute.* Cryptology ePrint Archive, Report 2006/105, 2006.

[5] J. Liang and X. Lai. *Improved Collision Attack on Hash Function MD5.* Cryptology ePrint Archive, Report 2005/425, November 2005.

[6] J. Reichardt and B. Schwarz. *VHDL-Synthese.* Oldenbourg, third edition, 2003.

[7] R. Rivest. *The MD5 Message-Digest Algorithm, RFC 1321*, 1992.

[8] M. Stevens. *Fast Collision Attack on MD5.* Cryptology ePrint Archive, Report 2006/104, 2006.

[9] X. Wang, A. Yao, and F. Yao. *Cryptanalysis of SHA-1.* Presented at the Cryptographic Hash Workshop hosted by NIST, October 2005.

[10] X. Wang, Y. L. Yin, and X. Yu. *Finding Collisions in the Full SHA-1.* In *Advances in Cryptology — EUROCRYPT 2005.* Springer Verlag, August 2005.

[11] X. Wang and X. Yu. *How to Break MD5 and other Hash Functions.* In *Advances in Cryptology — EUROCRYPT 2005.* Springer Verlag, May 2005.

Group Signatures With Observers Revisited

Thomas Schwarzpaul

Mathematical Institute, University of Giessen, Germany
Thomas.Schwarzpaul@math.uni-giessen.de

Abstract

A group signature scheme with observers was proposed in [KPW97]. In this paper, we show that the system is insecure in the sense of their definition. We also discuss the drawbacks of the system and show how these drawbacks can be minimized by presenting a new technique to divide the group's signing key using observers. Applying this technique to the offline electronic cash system proposed by Stefan Brands [Bra94] we present a new group signature scheme with observers.

Keywords. group signatures, blind signatures, observer, tamper-resistance device

1 Introduction

In 1991 *group signature schemes* were introduced by Chaum and van Heyst [CvH91]. In a group signature scheme a group member can create *anonymous* and *unlinkable* signatures on behalf of the group. By verifying a signature the verifier does not learn any information about the identity of the group member that created the signature. However, in case of misues or suspicion, the group signature can be *traced back* to the signing group member by a *trusted third party*.

In the last years many group signature schemes have been proposed and several applications for group signatures were suggested. In particular group signature schemes [BM01] have been introduced into the area of *electronic coin systems*.

In 1997 Kim et. al. [KPW97] introduced the idea of *group signatures with observers*. In the setting of group signatures an observer is a tamper-resistant device issued to all group members. The concept of *observers* was first introduced by Chaum and Pedersen [CP93] in the context of offline electronic coin systems. The approach of [KPW97] is especially interesting in combination with offline electronic coin systems where observers are also used. Unfortunately the protocols presented by Kim et. al. have some weaknesses. In this paper we identify the weaknesses of [KPW97] and present new protocols to solve the addressed weaknesses.

The rest of the paper is organized as follows: In section 2 we recall the group signature presented in [KPW97] and discuss the security definition. In section 3 we present a new technique to devide the group's signing key into two parts based on Shamir's secret sharing scheme [Sha79]. Finally we apply this technique to design a new group signature that features observers.

2 Group Signatures with Observers

In this section we review the system proposed by Kim et. al. [KPW97] and discuss its security.

2.1 The System of Kim et. al.

Before we start with a brief description of the system we explain the basic idea of the system: To setup the system the group manager GM selects a ordinary signature scheme and generates the group secret and public key. If a user U wants to join the group, GM issues an observer Oto U. The group's secret key is stored on the observer O. To sign a document U sends the document to O which signs the document under the group secret key. We now give a brief description of [KPW97].

Setup of the System. To setup the system the following parameters are chosen by GM

1. A ordinaray signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$,
2. The group's signing key X of Σ. The corresponding public key Y is published.
3. A probabilistic public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.
4. GM's key pair (E, D) of Π. The encryption key E is made public.

Registration. In order to registrate as a group member the following protocol is executed:

Step 1: U chooses a random secret key X_U and computes a public key Y_U corresponding to X_U. U sends Y_U to the group manager GM.

Step 2: GM keeps (U, Y_U) in his database, stores the common group secret key X and U's public key Y_U in U's tamper-resistant device O and hands O over to U.

Signature Generation. To sign a message m on behalf of the group the following protocol is executed between U and O:

Step 1: U computes a signature on the message m, $v = \mathsf{Sign}(X_U, m)$ and sends it to O.

Step 2: O verifies v with U's public key and computes the ciphertext $d = \mathsf{Enc}(E, v)$. O signs m and d with the group secret key $X, \sigma = \mathsf{Sign}(X, m||d)$ and sends $(m||d, \sigma)$ to U.

Step 3: U verifies $(m||d, \sigma)$. The group signature is the pair $(m||d, \sigma)$.

The group signature $(m||d, \sigma)$ is valid if $\mathsf{Vrfy}(Y, (m||d, \sigma))$ outputs true.

Tracing. To identify the signer of $(m||d, z)$, the group manager GM decrypts $v = \mathsf{Dec}(D, d)$ and find U which meets $\mathsf{Vrfy}(Y_U, (m, v)) = \mathsf{true}$.

2.2 Security Analysis

We now recall the security definition of group signatures with observers given by [KPW97] and argue that the system does not achieve security if the underlying signature scheme is not secure against a known message attack. According to [KPW97] a group signature scheme with observers is secure if the following definition holds:

Definition 2.1 *A group signature scheme must satisfy the following properties:*

1. *Unforgeability: Only group members, valid* U *and* O *are able to correctly sign messages.*
2. *Anonymity & Unlinkability: It is neither possible to find out which group member signed a message nor to decide wether two signatures have been issued by the same* U *and* O.
3. *Security against Framing attacks:*
   (a) *No matter how* U *deviates from the prescribed protocol, if* O *follows the protocol,* U *can neither circumvent the opening of a signature nor sign on behalf of other group members.*
   (b) *No matter how* O *and the organization deviate from the prescribed protocol, if* U *follows the protocol, the organization cannot correctly sign messages under* U*'s identity.*
   (c) *Even the deviated* (U, O) *cannot sign on behalf of other honest* $\tilde{\mathsf{U}}$*s.*

It is obvious that the security of the system only depends on the security of the underlying signature scheme Σ. If Σ is existentially forgeable under a known signature attack then the following attacks can be exexuted:

1. An attacker $\mathcal{A}$ can forge group signatures that cannot be traced.
2. U can generate valid signatures, where he can circumvent the opening of the signature.
3. U can generate signatures in another $\tilde{\mathsf{U}}$'s name, if he knows a signature generate by another user $\tilde{\mathsf{U}}$

To prevent these attacks Σ should be chosen as a signature scheme, which is not existentially forgeable under an adaptive chosen message attack.

Another drawback of the system is the fact that all computations are made by the tamper-resistant device O. From the group manager's point of view this is a problem because the signing key X is stored on O. Once just a single tamper-resistant device is compromised all security properties given in Definition 2.1 are broken. From the user's point of view this is undiserable since it only provides a low level of privacy, as U has no control over the messages sent from O, as argued in [CP93]. To protect U's privacy the protocols should be designed like an *electronic wallet* to prevent *inflow* and *outflow* information. For further information on electronic wallets we refered to [CP93]. Also O is assumed to be a tamper-resistant device, only the group's signing key is stored in the tamper-resistant part of the device. Signatures created by O are stored in the accessible part of the memory. Hence signatures generated by U can be linked if GM or an attacker $\mathcal{A}$ gets access to U's O. Therefore we suggest that protocols should be designed in a way that the information stored on O is not sufficient to link signatures generated by U.

3 A Signature Scheme With Observers

From the above description of [KPW97], we have seen that the most critical drawback of [KPW97] is the fact that the entire group's signing key is stored on a tamper-resistant device O. Once O is compromised the entire group signature scheme is insecure. An attacker $\mathcal{A}$, who is in possesion of the group's signing key, can easily sign messages on behalf of the group, without being identified afterwards. He can also sign messages on behalf of a group member U if he knows U's public key. For this reason the group's signing key should not be entirely stored on O. In this section we present a protocol where the group signature is generated by U in cooperation with O and not only by O.

To give an idea how the protocols work, we start with the idea behind the registration protocol: Let X be the group's signing key. If user U wants to join the group, GM randomly picks a number a and computes the polynomial $f = a \cdot x + X$. Notice that $f(0) = X$. GM then randomly picks x_1 and x_2 and computes $f(x_1), f(x_2)$. One of these points is stored on U's observer O, the other is sent to U. Now U and O can rebuild the group's signing key by Lagrange interpolation. Notice that this can be done in a way that neither U nor O have to reveal their secrets.

We now give a detailed description of how this idea can be applied to the Schnorr signature scheme [Sch91]. The signature scheme in section 2 can be replaced by the following signature scheme in order to minimize the threat of compromissing the observer. This is a first step to solve the weakness of [KPW97] adressed in section 2.2. We stress that this replacement does not solve the problem of inflow respectively outflow information in the group signatures with observers model of [KPW97]. But the following signature scheme can be used for anonymous group membership authentication.

Setup of System. To setup the system the group manager GM chooses two prime numbers p and q such that $q|p-1$. He determines an element $g \in \mathbb{Z}_p^*$ of order q, randomly picks $X \in_{\mathcal{R}} \mathbb{Z}_q$ and computes $y = g^X$. The group manager also chooses a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$. The parameters $(p, q, g, \mathcal{H})$ are made public, X is kept secret as the group's signing key. If not otherwise mentioned all computations in this section are done modulo p.

Registration. For each user U who wants to join the group the following protocol between U and the group manager GM is performed:

Step 1: The group manager GM randomly picks $a \in_{\mathcal{R}} \mathbb{Z}_q$ and determines $f = a \cdot x + X$. Notice that $f(0) = X$.

Step 2: GM randomly picks x_1, x_2 and computes $f(x_1), f(x_2)$.

Step 3: User U recieves $(x_2, f(x_2)), x_1$ and $y_1 = g^{f(x_1)}$.

Step 4: GM stores $(x_1, f(x_1))$ and x_2 on the observer O and sends O to U.

Signature Generation. If a user U wants to sign a document on behalf of the group the following protocol betwenn U and O is performed:

Step 1: O randomly picks $k_1 \in_{\mathcal{R}} \mathbb{Z}_q$, computes $r_1 = g^{k_1}$ and sends r_1 to U.

Step 2: U randomly chooses $s, k_2 \in \mathbb{Z}_q$, computes $r_2 = g^{k_2}$, $r = r_1 \cdot r_2$, as well as $c = \mathcal{H}(m, r)$ and sends c to O.

Step 3: O computes $d_1 = f(x_1) \cdot \frac{-x_2}{x_1 - x_2} \cdot c + k_1$ and sends d_1 to U.

Step 4: U verifies the following equation: $g^{d_1} = y_1^{\frac{-x_2}{x_1 - x_2} \cdot c} \cdot r$. If the equation holds Ucomputes $d_2 = f(x_2) \cdot \frac{-x_1}{x_2 - x_1} \cdot c + k_2$ and $d = d_1 + d_2$.

Security. Applying the forking lemma [PS00] and using the oracle replay attack we can prove the following result:

Theorem 3.1 *In the random oracle modell the above signature scheme is not existentially forgeable under a adaptive chosen message attack, even if the attacker knows the secret of the observer.*

We omit the proof due to the lack of space.

4 A new Group Signature Scheme With Observers

Based on the electronic cash system [Bra94] proposed by Stefan Brands, we present a new group signature scheme with observers. Due to the lack of space we omit the setup of the new system. For details we refer to [Bra94]. Besides the group manager GM and the group members a revocation manager RM participates in the system. The revocation manager RM randomly selects $X_{RM} \in_{\mathcal{R}} \mathbb{Z}_q$ and publishes $g_2^{X_{RM}}$ as his public key of an ElGamal encryption scheme. If not otherwise mentioned all computations in this section are done modulo p.

Registration. Besides the procedure presented in section 3 this additional step is performed between U and GM:

U picks $u_1 \in_{\mathcal{R}} \mathbb{Z}_q$ and computes $I = g_1^{u_1}$. U sends I to GM and proves that he knows the dicrete logarithm of I with respect to g_1. GM stores I in his database and on O. Finallay GM issues O to U. The number I is U's unique identiy, which is encrypted under the public key of RM during the signature generation, executed between U and O.

Signature Generation.

Step 1: O randomly picks $\omega \in_{\mathcal{R}} \mathbb{Z}_q^*$, computes $a_1 = g_1^{\omega}, b_1 = I^{\omega}, b_2 = g_2^{\omega}$ and sends a_1, b_1, b_2 to U.

Step 2: U randomly chooses $s, u, v, x_1, x_2 \in_{\mathcal{R}} \mathbb{Z}_q^*$ and computes the following: $A_1 = I g_2^s$, $A_2 = f_2^s$, $B = g_1^{x_1} g_2^{x_2}$, $\tilde{A}_2 = f_2^{x_2}$, $a = a_1^u g^v$, $b = b_1^u b_2^{su} A^v$, $z = h_1^s h_2$, $c = \mathcal{H}(A_1, A_2, B, \tilde{A}_2, z, a, b)$, $c_1 = c/u \bmod q$. User U sends c_1 to O.

Step 3: O computes $r' = c_1 \cdot \frac{-x2}{x_1 - x_2} \cdot f(x_1) + \omega \bmod q$ and sends r' to U.

Step 4: Ucomputes $r = c_1 \cdot u \cdot \frac{-x_1}{x_2 - x_1} f(x_2) + r' \cdot u + v \bmod q$ and $r_1 = c \cdot u_1 + x_1 \bmod q$, $r_2 = c \cdot s + x_2 \bmod q$.

The signature on the message m is the tupel $(A_1, A_2, B, \tilde{A}_2, a, b, z, c, r, r_1, r_2)$. To verify a signature the verifier checks the following equations: $g^r = h^c a$, $A_1^r = z^c b$, $g_1^{r_1} g_2^{r_2} = A_1^c B$ and $f_2^{r_2} = A_2^d \tilde{A}_2$.

Tracing. To open a group signature the revocation manager RM decrypts the ElGamal encryption (A_1, A_2).

Security. Due to the lack of space we omit the proof of the follwing theorem.

Theorem 4.1 *Under the assumption that Brand's blind signature scheme is a restrictive blind signature scheme the presented group signature scheme is secure in the sense of definition 2.1 and prevents inflow respectively outflow information.*

5 Conclusions

We argued that the system proposed by [KPW97] does not achieve security in terms of their definition, if a signature scheme is used, which is existentially forgeable under a known signature attack. Further we showed how to minimize the drawbacks of [KPW97] by presenting a new signature scheme with observers. Applying the technique used for the new signature scheme with observers, we presented a new group signature scheme with observers based on the offline electronic cash system [Bra94] proposed by Stefan Brands.

References

[BM01] Colin Boyd and Greg Maitland. Fair electronic cash based on a group signature scheme. In *Proceedings Of ICICS '01*, volume 2229 of *Lecture Notes in Computer Science*, pages 461–465. Springer, 2001.

[Bra94] Stefan Brands. Untraceable offline cash in wallets with observers. In *Advances In Cryptology - Proceedings Of Crypto '93*, volume 740 of *Lecture Notes in Computer Science*, pages 302–318. Springer, 1994.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Advances In Cryptology - Proceedings Of Crypto '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1993.

[CvH91] David Chaum and Eugène van Heijst. Group signatures. In *Advances In Cryptology - Proceedings Of Eurorypt '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.

[KPW97] Seungjoo Kim, Sangjoon Park, and Dongho Won. Group signatures with observers. In *Proceedings Of CISCS '97 - Conference on Information Security and Cryptology*, pages 19–25, 1997.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal Of Cryptology*, 13(3):361–396, 2000.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal Of Cryptology*, 4(3):161–174, 1991.

[Sha79] Adi Shamir. How to share a secret. *Communications Of the ACM*, 22(11):612–613, 1979.

Modified HB Authentication Protocol

Selwyn Piramuthu, Yu-Ju Tu

Information Systems and Operations Management
University of Florida, Gainesville, FL 32611, USA
selwyn@ufl.edu,tuyuju@ufl.edu

Abstract

HB (Hopper and Blum, 2001) use the hardness of learning parity with noise (LPN) problem to develop their authentication protocol. After it was shown to be vulnerable to attack by an adversary (e.g., Juels and Weis, 2005), several modifications were proposed (e.g., Bringer, et al., 2006; Duc and Kim, 2007; Munilla and Peinado, 2007). Most of these have been shown to still be vulnerable to attacks by an active adversary. We propose a modified HB protocol.

Keywords. RFID, Authentication Protocols, HB

1 Introduction and Background

HB (Hopper and Blum, 2001) is a multiple-round protocol for authenticating single RFID tag/reader combination. It was shown to be as hard as the learning parity with noise (LPN) problem. HB uses a single shared secret key between reader and tag. HB was shown to be vulnerable to attack by an adversary (e.g., Juels and Weis, 2005). Given the strength of the underlying LPN problem, several modifications to HB have been proposed (e.g., Bringer, et al., 2006; Duc and Kim, 2007; Munilla and Peinado, 2007) soon thereafter.

Juels and Weis (2005) proposed HB$^+$ with two secret keys and proved that it was secure against attacks by an adversary. However, Gilbert et al. (2005) showed that HB$^+$ was not secure against attacks by an active adversary. Bringer et al. (2006) proposed further modifications to HB$^+$ using the same two shared secret keys between tag and reader, but with the addition of a rotating function. Since it shares some common structure with HB$^+$, it too was shown to be vulnerable to attacks (Piramuthu, 2006). Recently, Munilla and Peinado (2007) and Duc and Kim (2007) proposed modified protocols that are claimed to be secure against attacks. We show that these are vulnerable to attacks in the next section. In Section 3, we present the proposed modified HB protocol.

2 Two Recently Proposed HB-related Protocols

2.1 HB-MP

Munilla and Peinado (2007) present two protocols that are variants of HB. The first one is presented in Figure 1. Here, it is easy to see that an adversary impersonating a valid tag can send r_A to the reader and the authentication process will trivially succeed. They propose another protocol in which they use two secrets (x and y) and compute $x \leftarrow$ Rotate(x, y_i) and $z = r_A.xm \oplus \nu$ where xm is the m-bit binary vector with the m least significant bits of x and Rotate(x, y_i) is a bit-wise left rotate operator that rotates x by y_i positions. Here, too, an adversary impersonating a tag can just send r_A to the reader for the authentication process to be successful

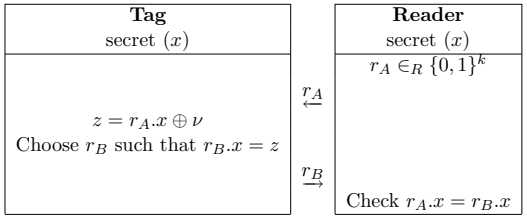


Figure 1: A round of HB-MP$\prime$ (Munilla and Peinado, 2007)

2.2 Protocols due to Duc and Kim (2007)

Duc and Kim present a variation of HB$^+$ protocol that is resistant to Gilbert et al.'s attack (Figure 2). They use an additional secret (s) as well as additional noise parameter ($\eta\prime$). The main difference between HB$^+$ and HB* is in the way in which the checking function is computed. Duc and Kim switch x and y in the computed function depending on a boolean variable r. They claim that this prevents Gilbert et al.'s attack. However, an adversary impersonating a reader can send r_B (instead of r_A) to the tag, which then computes z. Except, r is made useless in this scenario and, $z = r_B.(x \oplus y)$. Knowing r_B, the adversary can now learn $x \oplus y$. Whenever $z = 1$ and $r_B = 1$, $x \oplus y = 1$. And, whenever $z = 0$ and $r_B = 1$, $x \oplus y = 0$. The adversary can use this information to track the tag.

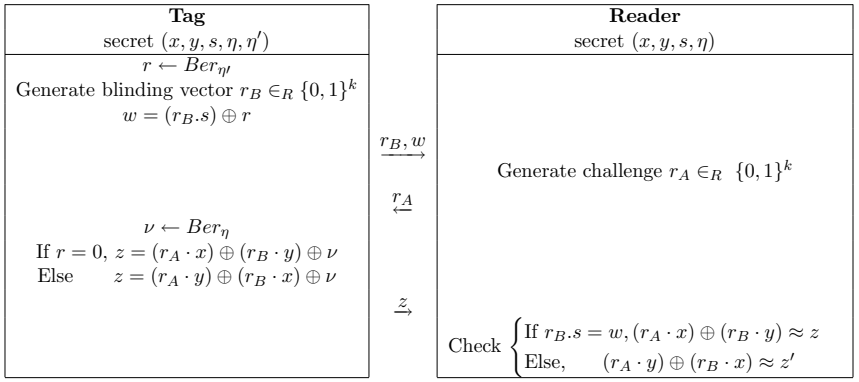


Figure 2: HB* Protocol due to Duc and Kim (2007)

3 Proposed Authentication Protocol

This protocol (Figure 3) is more realistic compared to HB$^+$ since the reader, as opposed to the tag, initiates the authentication process. Similar to HB$^+$, there are two shared secrets (x, y) and a fresh nonce is generated by tag and reader in each round to prevent replay attack. The reader-generated nonce (r_B) is XOR'd with x to make it harder for the adversary to know x

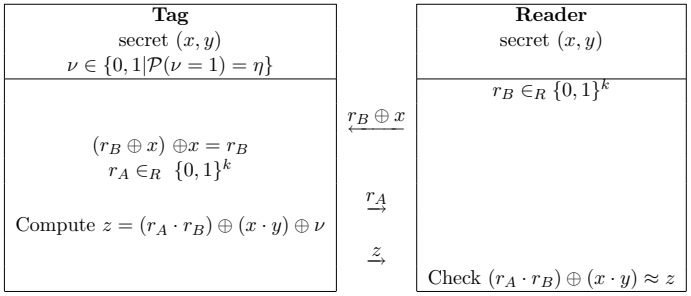


Figure 3: A round of modified HB$^+$ protocol

and/or r_B. The nonces are not (scalar) multiplied with the secrets to prevent an adversary from mounting a Gilbert et al.-style attack and identifying one of the secrets. In fact, Gilbert et al.'s man-in-the-middle attack would reveal r_A, but this is not an issue since it is transmitted in plain text format and the protocol seems to be secure in spite of it.

3.1 Security Analysis of Modified Protocol

We consider the modified protocol presented in Figure 3, and evaluate its security against common threats.

Attack on a tag. An adversary impersonating a valid reader can modify the only message sent from reader to tag ($r_B \oplus x$). Since the tag freshly generates nonce r_A only after receiving ($r_B \oplus x$) from the reader, replaying this message will not result in decipherable response from the tag due to the secret (y) and r_A.

Attack on the reader. An adversary impersonating a tag can modify r_A and/or z. This type of attack will not succeed because of the shared secret keys (x and y).

Attack on the communication between tag and reader. An adversary can block messages between the reader and a tag. When this happens, the authentication process is broken and it doesn't succeed. However, since keys are not updated during every round, an incomplete authentication round will not affect future authentication processes.

Attack on user privacy. Since no 'private' information is transmitted in plain text during validation, this is of no concern here.

Attack on location privacy. This is of concern since the secrets do not change over different runs of the protocol. Once the secrets are known, it is relatively easy to track these tags and readers.

Attack against the key. This happens when an attacker listens in on the transaction and tries to identify the key values. Again, if the keys are selected appropriately (e.g., Lenstra and Verheul, 2001), this is not of concern.

Attack against implementation. Provided the keys and the random numbers are genereted with caution, this is not of concern.

Disassembling the tags. These tags are clearly not tamper-resistant, and can be disassembled to retrieve the structure of z as well as the secret keys (x, y).

4 Discussion

We considered multiple round protocols for single tag based on the hardness of LPN problem. Although the underlying basis is strong, the protocols in this category have been shown to be

vulnerable for relatively short keys that are commonly used in low-cost passive tags. We identify additional vulnerabilities in existing protocols in this stream and propose a modified version that addresses these vulnerabilities. We were unable to identify any vulnerability in the proposed protocol, although such protocols are secure only until someone identifies vulnerabilities. We, therefore, submit our protocol to the scrutiny of researchers in this area to identify inherent vulnerabilities.

References

[1] J. Bringer, H. Chabanne, and E. Dottax. "HB^{++}: a Lightweight Authentication Protocol Secure Against Some Attacks," *IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU*, 2006.

[2] D.N. Duc, and K. Kim. "Securing HB$^+$ against GRS Man-in-the-Middle Attack," *Proceedings of the Symposium on Cryptography and Information Security (SCIS2007)*, 2007.

[3] H. Gilbert, M. Robshaw, and H. Sibert. "An Active Attack Against HB$^+$ - A Provably Secure Lightweight Protocol," *IEE Electronic Letters*, 41, 21, 2005, pp. 1169-1170.

[4] N.J. Hopper and M.M. Blum. "Secure Human Identification Protocols." in C. Boyd (ed.) *Advances in Cryptology - ASIACRYPT 2001*, Volume 2248, *Lecture Notes in Computer Science*, 2001, pp. 52-66, Springer-Verlag.

[5] A. Juels and S.A. Weis. "Authenticating Pervasive Devices with Human Protocols," in V. Shoup (ed.) *Advanced in Cryptology - CRYPTO'05*, Volume 3126, *Lecture Notes in Computer Science*, 2005, pp. 293-308, Springer-Verlag.

[6] A.K. Lenstra and E.R. Verheul. "Selecting Cryptographic Key Sizes," *Journal of Cryptography*, 14(4), 2001, pp. 255-293.

[7] J. Munilla and A. Peinado. "HB-MP: A Further Step in the HB-family of Lightweight Authentication Protocols," *Computer Networks*, 51(9), June 2007, pp. 2262-2267.

[8] S. Piramuthu. "HB and Related Lightweight Authentication Protocols for Secure RFID Tag/Reader Authentication," *CollECTeR Europe Conference*, Basel, 2006.

Cryptography Based on Quadratic Forms: Complexity Considerations

Rupert J. Hartung

Johann Wolfgang Goethe Universität Frankfurt a. M.
Postfach 11 19 32; Fach 238
60054 Frankfurt a. M., Germany
http://www.mi.informatik.uni-frankfurt.de
hartung@mi.informatik.uni-frankfurt.de

Abstract

Motivated by a recent identification scheme based on indefinite quadratic forms, we analyze the computational problem it is based on, which is essentially the problem of finding an integral change of basis transforming equivalent forms into each other. It turns out that we can relate it to an NP-hard problem fo a fixed number of variables, under reasonable number-theoretic heuristics. Moreover, we confirm unconditionally that the complexity of the problem is concentrated in fixed dimension, and compare the transformation problem to factoring.

Keywords. quadratic form, complexity, NP-complete, Cohen-Lenstra heuristics, identification scheme

1 Introduction

Lattice-based cryptography has been a vivid field in cryptologic research ever since its proposal (see [AD97], [HPS98], [HPS01], [HHGP+03], [GGH97]). A main advantage of this family of cryptosystems is considered to be the fact that the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP) have been proven to be NP-complete (partially under randomized reductions, see [MG02], [Kho05]). This is taken as a hint that this type of primitives may still be secure and applicable in the (still hypothetical) age of quantum computers because cryptographic protocols based on NP-hard problems seem to withstand attacks by quantum computers (see [BBBV97]).

However, lattice cryptography intrinsically requires lattices of high dimension, which lead to large cryptographic keys and slow down the protocols. By contrast, this problem is overcome by the use of indefinite quadratic forms.

A *quadratic form* (or simply *form*) is a homogeneous polynomial of degree 2 over some unique factorization domain R, which we assume not to be of characteristic 2. *i.e.*$f = x^t Ax$ with $A \in \frac{1}{2}R^{n\times n}$ symmetric, with integral principal diagonal, and an indeterminate vector $x = (x_1, \ldots, x_n)^t$. n is called the *dimension*, and $\det f := \det A$ the *determinant* of the quadratic form f. For $T \in R^{n\times n}$ let fT denote the form $x^t T^t ATx$. Two forms f, g are called *equivalent* over R if there is $T \in \mathrm{GL}_n R$ such that $g = fT$; notation $f \sim_R fT$. The equivalence class of f is simply called the *class* of f.

In [HS07], an identification scheme has been proposed according to the following outline:

Public key: Equivalent forms f_0, f_1,
Secret key: $S \in \mathrm{GL}_n R$ such that $f_0 S = f_1$.

1. Prover $\mathcal{P}$ picks $T \in R^{n\times n}$ (according to some distribution) ;
computes $g := f_0 T$, and sends g,

2. Verifier $\mathcal{V}$ sends a random one-bit challenge $b \in_R \{0, 1\}$,
3. $\mathcal{P}$ sends $Q := S^{-b}T$, and $\mathcal{V}$ checks that $f_b Q = g$.

Remark. The concrete proposal of [HS07] uses the concept of LLL-reduction for indefinite quadratic forms (see [Sim05] and [IAS96]) as a key ingredient to smoothen the employed probability distributions. It is shown in [HS07] that this scheme is complete, a proof of knowledge, and statistically zero-knowledge for $R = \mathbb{Z}$, $n = 3$ under some additional hypothesis on the distribution in question.

In this paper, we shall ignore issues about random distributions and concentrate on how the complexity of the computational problem behind changes with the use of different rings and families of quadratic forms. We will come to the conclusion that the choice $R = \mathbb{Z}$, $n = 3$ is likely to be "right". Obviously, the basic problem of extracting the secret key from the public key is the following:

> **Trafo$^R(\mathcal{P})$ The transformation problem over R**
> *PARAMETERS:* Set $\mathcal{P}$ of properties of quadratic forms, domain R.
> *INPUT:* quadratic forms f, g of dimension satifying all properties from $\mathcal{P}$ and $f \sim_R g$.
> *OUTPUT:* $S \in \mathrm{GL}_n\mathbb{Z}$ such that $g = f\,S$ (where n is the dimension of f).

We have been a bit vague on the set $\mathcal{P}$. For all our applications, however, we use natural properties (*e.g.*on the dimension) which can be encoded in an essentially canonical way.

Note that we do not insist that a specific S should be constructed as any such matrix would help to pass the scheme.

As a case of comparison and because of its simplicity, we also consider the problem of solving homogeneous quadratic equations: A quadratic form f is said to *represent* $m \in R$ if there is $v \in R^n \setminus \{0\}$ such that $f(v) = m$. It is said to represent m *primitively* if v can be chosen primitive, *i.e.*$\gcd(v_1, \ldots, v_n) = 1$.

> **Repr$^R(\mathcal{P})$ The representation problem over R**
> *PARAMETERS:* Set $\mathcal{P}$ of properties of quadratic forms, domain R.
> *INPUT:* a quadratic form f satifying all properties from $\mathcal{P}$, and $m \in R$ such that f represents m primitively.
> *OUTPUT:* a primitive vector $u \in R^n$ such that $f(u) = m$ (where n is the dimension of f).

A quadratic form f (over $\mathbb{Q}$ or $\mathbb{Z}$) is called *indefinite* if it represents both positive and negative numbers, and *definite* if it represents either only positive or only negative numbers. Lattices are closely related to positive definite forms. It is called *isotropic* if it represents 0, and anisotropic otherwise.

Note that for fixed dimension, the transformation problem for definite forms can be solved in polynomial-time (although the complexity seems to be devastating as the dimension increases), see[PP85], [PS85]. We shall use this fact later on.

2 Choice of the base ring

A natural candidate for R, apart from the integers, are fields. For finite fields $\mathbb{F}_q$, however, there are efficient (probabilistic) algorithms for homogeneous quadratic equations (see [PS87], [Cas78, ch. 2]), which already rules them out for our applications.

For the field of rational numbers, however, we observe that solving the transformation problem is essentially equivalent with factoring integers. This is made precise by the following theorem.

Denote by $\preccurlyeq$ polynomial-time reducibility and by $\preccurlyeq_r$ random polynomial-time reducibility of computational problems.

Theorem 2.1 *Let $\mathcal{P}$ consist of the properties of being indefinite, anisotropic, and of dimension 3.*

(a) Denote by **Sqrt** *the problem of computing a square root of -1 mod an integer N, existence supposed. Then* $\mathbf{Sqrt} \preceq_r \mathbf{Trafo}^{\mathbb{Q}}(\mathcal{P})$. *One oracle call suffices.*

(b) Denote by **Fact** *the problem of factoring integers. Then* $\mathbf{Trafo}^{\mathbb{Q}} \preceq \mathbf{Fact}$. *For an instance (f, g) of* **Trafo**, *it suffices to call the oracle once to factor* $(\det f)(\det g)$.

As there is no essientially better algorithm know to extract modular square root than by the aid of factoring the modulus, we may informally state that the transformation problem is 'almost' equivalent to factoring.

Proof.

(a) Let N be an instance of **Sqrt**, *i.e.*N contains only prime factors $p \equiv 1 \mod 4$. Choose $a \in \mathbb{Z}$ which is a quadratic non-residue mod N $f := \langle 1, -N, aN\rangle, \qquad g := \langle -1, N, aN\rangle$. It can be computed from the Hasse principle (see [Cas78]) that $f \sim_{\mathbb{Q}} g$ and that f, g are anisotropic because of the condition on the prime factors on N. Now any $S = (s_{ij}) \in \mathrm{GL}_n\mathbb{Q}$ with $S^t A_f S = A_g$ satisfies $(s_{11}k)^2 = -1 \mod N$, where $k \in \mathbb{Z}$ is such that $kS \in \mathbb{Z}^{n\times n}$.

(b) *(Sketch)* This is essentially an algorithmic version of the classical proof of the weak Hasse principle by the strong Hasse principle, see again [Cas78]. We employ Simon's algorithm [Sim05] to find isotropic vectors of the form $f(x) - g(y)$. The factorization of the determinants is required in this routine.

□

As we are on the look-out for problems much harder than factoring, we avoid the field $\mathbb{Q}$ and consider forms over $\mathbb{Z}$.

3 Concentration in dimensions 3,4

Motivated by the last section, we now analyse the complexity of the transformation problem if the factorization of the determinants are already given. Denote this variant of the problem by **FTrafo** instead of **Trafo**, and analogously for **FRepr**. From now on, we drop the superscript R, as we restrict ourselves to $R = \mathbb{Z}$.

Theorem 3.1 *Denote by $\mathcal{F}_n(d)$ the properties* $\det f = d$ *and* $\dim f = n$ *for a quadratic form f.*

Let $n \geq 5$. let $d \in \mathbb{Z}$ be odd and squarefree and let the factorization of d be given. Then

$$\mathbf{FTrafo}(\mathcal{F}_n(d)) \preceq \mathbf{FTrafo}(\mathcal{F}_{n-2}(d))$$

This has the following striking consequence.

Corollary 3.2 *Let $\mathcal{F}$ denote the properties "*$\det f$ *is odd and squarefree" and "*$\dim f \geq 3$*" for a quadratic form f, and $\mathcal{F}_{3,4}$ for additionally "*$\dim f \in \{3, 4\}$*". Then*

$$\mathbf{FTrafo}(\mathcal{F}) \preccurlyeq \mathbf{FTrafo}(\mathcal{F}_{3,4})$$

□

In the proof, we will consider equivalence over local rings $\mathbb{Z}_p$, with p a prime or $\mathbb{Z}_\infty = \mathbb{R}$ (see [Cas78], [O'M63]). f, g are said to belong to the same *genus* if $f \sim_{\mathbb{Z}_p} g$ for all symbols p. Obviously, this is necessary for $f \sim_{\mathbb{Z}} g$ (from now on denoted by $f \sim g$).

Proof. *(of theorem.)* Let (f, g) be an instance of $\mathbf{Trafo}_n(d)$. By Meyer's theorem [Cas78], f is either definite or isotropic since $n \geq 5$. As noted in the introduction, for definite forms the problem can be efficiently solved, so assume indefiniteness. As d is squarefree, f can be efficiently transformed into $f\,S_1$ with matrix of the shape

$$\begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & A_0 \end{pmatrix}$$

for some matrix A_0 of an $(n-2)$-dimensional form f_0 by [Sch04] (using the factorization of $\det f$), and analogously g_0 for g. Now by Witt's lemma for p-adic integers [Cas78], $f_0 \sim_{\mathbb{Z}_p} g_0$ for all symbols $p \neq 2$. Since d is odd, it follows that f_0 and g_0 belong to the same genus. But by [Cas78, p. 202f.], we deduce that $f \sim g$ because $\dim f_0 \geq 3$ and d is squarefree. Obviously, S_0 with $f_0 S_0 = g_0$ can be extended to a matrix solving the original problem. □

4 NP-Hardness

In [HS07], it is proved that a decisional variant of the representation and transformation problems are NP-complete under randomized reductions, conditional to what we call the special Cohen-Lenstra heuristics. More precisely, we introduce the following variant of **Trafo**:

DITrafo Descisional Interval Transformation Problem
PARAMETERS: Set $\mathcal{P}$ of properties of quadratic forms.
INPUT: $n \in \mathbb{N}$, n-ary quadratic forms f, g satifying all properties from $\mathcal{P}$, matrices $A, B \in (\mathbb{Z} \cup \{\pm\infty\})^{n\times n}$, factorization of $\det f$.
DECIDE: Whether there is $T \in \mathrm{GL}_n(\mathbb{Z})$, $A_{ij} \leq T_{ij} \leq B_{ij}$ for all i, j such that $f\,T = g$.

Analogously this can be done for the representation problem. Note that **DITrafo** is polynomial-time equivalent with actually computing a transformation with coefficients in the given intervals, by a straightforward divide-and-conquer algorithm.

Recall that the complexity class RP (random polynomial time) consists of all descicion problems for which there is a probabilistic worst-case polynomial-time algorithm which accepts any 'yes'-instances with probability $\geq \frac{1}{2}$, and rejects every 'no'-instance.

Theorem 4.1 *Let $M \in \mathbb{N}$. Let $\mathcal{P}$ consist of the properties* $\dim f = 3$, *and f indefinite anisotropic for a quadratic form f. If the special Cohen-Lenstra heuristics holds true, then* $\mathbf{DIRepr}(\mathcal{P})$ *and* $\mathbf{DITrafo}(\mathcal{P})$ *are NP-complete under randomized reductions with one-sided error; more precisely:*

$$NP \subseteq RP^{\mathbf{DIRepr}(\mathcal{P})}, \qquad NP \subseteq RP^{\mathbf{DITrafo}(\mathcal{P}')}.$$

For details, see [HS07].

References

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the 29th annual ACM symposium*

on theory of computing, El Paso, TX, USA, May 4-6, 1997, pages 284–293, New York, 1997. Association for Computing Machinery.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal of Computing*, 26(5):1510–1523, 1997.

[Cas78] John W.S. Cassels. *Rational Quadratic Forms*. Number 13 in London Mathematical Society Monographs. Academic Press, 1978.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. jun. Kaliski, editor, *Advances in Cryptology - CRYPTO '97, 17th annual international cryptology conference. Santa Barbara, CA, USA.*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer-Verlag, 1997.

[HHGP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSign: Digital signatures using the NTRU lattice. In Marc Joye, editor, *Topics in cryptology – CT-RSA 2003. The cryptographers' track at the RSA conference 2003, San Francisco, CA, USA, April 13–17, 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 122–140. Springer-Verlag, 2003.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In J. P. Buhler, editor, *Algorithmic number theory. 3rd international symposium, ANTS-III, Portland, OR, USA, June 21–25, 1998*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer-Verlag, 1998.

[HPS01] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NSS: an NTRU lattice-based signature scheme. In Birgit Pfitzmann, editor, *Advances in cryptology - EUROCRYPT 2001. 20th international conference on theory and application of cryptographic techniques, Innsbruck, Austria, May 6-10, 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 211–228. Springer-Verlag, 2001.

[HS07] Rupert Josef Hartung and Claus-Peter Schnorr. Public key identification based on the equivalence of quadratic forms. preprint, 2007.

[IAS96] Gábor Ivanyos and Ágnes Szánto. Lattice basis reduction for indefinite forms and an application. *Journal on Discrete Mathematics*, 153(1–3):177–188, 1996.

[Kho05] Subash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM*, 52(5):789–808, 2005.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

[O'M63] O. T. O'Meara. *Introduction to Quadratic Forms*, volume 117 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1963. reprinted in 2000.

[PP85] W. Plesken and M. Pohst. Constructing integral lattices with prescribed minimum. I. *Mathematics of Computation*, 45:209–221, 1985.

[PS85] W. Plesken and B. Souvignier. Computing isometries of lattices. *Mathematics of Computation*, 45:209–221, 1985.

[PS87] John M. Pollard and Claus-Peter Schnorr. An efficient solution of the congruence $x^2 + ky^2 = m (mod\, n)$. *IEEE Transactions on Information Theory*, 33:702–709, 1987.

[Sch04] Claus-Peter Schnorr. Reduction of quadratic forms reconsidered. (preprint), 2004.

[Sim05] Denis Simon. Solving quadratic equations using reduced unimodular quadratic forms. *Mathematics of Computation*, 74(251):1531–1543, 2005.

Lattice-Based Computationally-Efficient Private Information Retrieval Protocol

Carlos Aguilar-Melchor and Philippe Gaborit

XLIM - Université de Limoges, France Carlos.Aguilar@xlim.fr,Philippe.Gaborit@xlim.fr

1 Introduction

A Private Information Retrieval (PIR) scheme is a protocol in which a user retrieves a record out of N from a replicated database, while hiding from the database which record has been retrieved, as long as the different replicas do not collude ([Ga]).

A specially interesting sub-field of research, called single-database PIR ([K0]), deals with the schemes that allow a user to retrieve privately an element of a non-replicated database. In these schemes, user privacy is related to the intractability of a mathematical problem, instead of based on the assumption that different replicas exist and do not collude against their users. There are wide applications for instance for medical files, video or songs databases or even stock echanges share prices, for all these cases one may want to retrieve information in a database without the database knowing which information is searched for.

The main performance measure used for these schemes is communication cost, disregarding computational cost. Thereby, current single-database PIR schemes provide almost optimal communication cost but require the database to use an enormous amount of computational power. This limits greatly the usability of these schemes, as even high-end servers are unable to generate PIR replies in a reasonable time for anything but the smallest databases.

In this work we present a lattice-based PIR scheme, using an NTRU-like approach, in which the computational cost is a few thousand bit-operations per bit in the database. This improves the protocol computational performance by two orders of magnitude when compared to existing approaches. Our scheme has communication performance not as good as other existing protocols, but eventually its far better computational performance permits to make our PIR protocol almost usable comparing to previously known protocols which are very difficult to use.

2 Description

2.1 Basics

Databases are usually described in PIR literature as a set of n bits among which a user wants to retrieve one. In the real world, a database is scarcely seen as a uniform set of bits, but rather as a set of n l-bit elements, and users want to retrieve a complete l-bit element. As we are interested in practical usage of PIR schemes, we will throughout this paper use this description instead of the one traditionally used in PIR literature.

We will not consider the distinction between users wanting to retrieve a single bit from a database and users wanting to retrieve a set of bits (operation usually called Private Block Retrieval). We suppose that users always want to retrieve l-bit elements from the database with $l \gg 1$. Usual PIR schemes proceed in the following way:

- a request is generated with values associated to each element of the database and a special value associated to the particular element to be retrieved.

-an answer is computed with a protocol relating a sequence of bits of each element of the database with the value of the query and then with other elements.

- the answer is sent back and it is possible to extract the bit sequence associated to the special element.

One searches for protocols which minimize the expansion factor (the ratio of the size of the answer with number of bits retrieved) and the computational cost (the cost of generating the answer). We refer to [Ga] for an extensive survey of the field.

2.2 High-level overview of our protocol

The PIR scheme we propose relies on the simple idea of controlled noise addition. The main idea is to to start from a secret random $[N, 2N]$ matrix M of rank N over a field Z/pZ. This matrix is used to generate a set of different matrices obtained by multiplication on the left side by invertible random matrices. These matrices (which can also be seen as lattices by joining pI_{2N} for I_{2N} the identity $2N \times 2N$ matrix) are disturbed by the user by the introduction of noise in half of the matrices' columns to obtain respectively softly disturbed matrices (SDMs) and hardly disturbed matrices (HDMs).

To obtain an element from the database the user sends a set of SDMs and one HDM. The database inserts each of its elements in the corresponding matrix with a multiplicative operation OP and sums all the rows of the resulting matrices to obtain the database reply, a single noisy vector. Using the unmodified columns of the matrices sent in the request, the user is able to find the noise associated to the returned noisy vector. If the soft noise multiplied by the total noise factor (which is proportional to the number of elements in the database) is much smaller than the hard noise, it can be filtered out and the user can retrieve the information associated to the noise of the HDM matrix. The scheme uses the same kind of idea that for the lattice-based NTRU cryptosystem: one considers a vector space over a field Z/pZ where the key idea is to control an error by keeping it non altered by any modular operation.

2.3 Request generation

The scheme will have three global integer parameters: $2N$, the dimension of the lattice and special parameters p and q. The database is described as a set of n l-bit elements, and we note i_0 the index of the database element the user is interested in. To obtain a PIR request, the user will follow:

Protocol 1

1. Note $l_0 = \lceil log(n \times N) \rceil + 1$ and set q as 2^{2l_0} and p as a prime larger than 2^{3l_0}.
2. Generate A and B, two random matrices over Z/pZ such that A is invertible, and note $M = [A|B]$.
3. For each $i \in \{1 \cdots N\}$ compute a matrix $M_i^{''} = [A_i|B_i]$ by multiplying M by a random invertible matrix P_i.
4. Generate the random scrambling matrix Δ as a $N \times N$ random diagonal matrix over Z/pZ.
5. For each $i \in \{1 \cdots N\} \backslash i_0$ generate the soft noise matrix D_i, a $N \times N$ random matrix over $\{-1, 0, 1\}$, and compute the soft disturbed matrix $M_i^{'} = [A_i|B_i + D_i\Delta]$.
6. Generate D_{i_0}, the hard noise matrix, by:

   - generating a soft noise matrix,
   - replacing each diagonal term by q.
7. Compute the hard disturbed matrix $M_{i_0}^{'} = [A_{i_0}|B_{i_0} + D_{i_0}\Delta]$.
8. Choose a random permutation of columns $\mathcal{P}(\cdot)$ and compute $M_i = \mathcal{P}(M_i^{'})$ for $i \in \{1 \cdots n\}$.
9. Send the ordered set $\{M_1 \cdots M_n\}$ to the database.

2.4 Answer encoding

To answer to the PIR reply the database follows protocol 2. The result is a vector V of dimension $2N$ over Z/pZ. We will suppose that each element has exactly the number of bits that can be encoded into a matrix ($l_0 \times N$ bits).

Protocol 2

1. Split each database element m_i in N l_0-bit integers $\{m_{i1} \cdots m_{iN}\}$.
2. For each $i \in \{1 \cdots n\}$ construct the vector $v_i = \sum_{j=1}^{N} m_{ij} M_{ij}$ where M_{ij} denotes the j-th row of M_i.
3. Return $V = \sum_{j=1}^{n} v_i$

2.5 Information extraction

To extract the information from the database reply, the user will operate in two phases. First he will recover the noise included in the vector (steps 1 and 2 of protocol 3, and then he will unscramble and filter out this noise to obtain the information (steps 3 to 5).

Protocol 3

1. Compute the non-permuted noisy vector $V^{'} = \mathcal{P}^{-1}(V)$.
2. Retrieve $E = V_D^{'} - V_U^{'} A^{-1} B$, the scrambled noise, $V_U^{'}$ and $V_D^{'}$ being resp. the undisturbed and disturbed halves of $V^{'}$.
3. Compute the unscrambled noise $E^{'} = E\Delta^{-1}$
4. For each $e_j^{'}$ in $E^{'} = [e_1^{'} \cdots e_n^{'}]$, compute $e_j^{''} = e_j^{'} - \epsilon$ with $\epsilon := e_j^{'} \% q$ if $e_j^{'} \% q < q/2$ and $\epsilon := e_j^{'} \% q - q$ else.
5. For each $j \in \{1 \cdots n\}$, compute $m_{i_0 j} = e_j^{''} q^{-1}$.

To recover the noise, the user will first undo the random column permutation (step 1). Then he will use the N first coordinates of the vector and the initial matrix M to obtain what the N last coordinates (which have been disturbed) should be without noise. He will use these values to extract the noise inserted in these coordinates (step 2).

This noise is composed of soft and hard noise, but it cannot be directly filtered because it was scaled up by the noise scrambling matrix. The user will therefore first eliminate this scrambling (step 3). He will then filter out the soft noise (step 4), and divide each coordinate by the hard noise factor to obtain the database sub-elements of m_{i_0}. For lack of space we do not develop here why this works.

3 Security of the scheme

3.1 A new problem: the Hidden Lattice Problem

The security of our scheme relies on a new lattice-based problem that we define:

Hidden Lattice Problem

Let V be a k dimensional vector space of length n over a finite field K. Consider a set of r different random basis $\{V_1 \cdots V_r\}$ with $V_i = [V_{i,1}| \cdots |V_{i,n}]$ of V. Fix randomly a subset of k independent columns from these basis and note $S = \{j_1 \cdots j_{n-k}\}$ its complementary set. For each V_i generate a set of random columns $\{R_{i,1} \cdots R_{i,n-k}\}$ of elements in $\{-1, 0, 1\}$. Disturb each V_i into V_i' by adding these random columns to $\{V_{i,j_1} \cdots V_{i,j_{n-k}}\}$. Deduce from the set of disturbed basis which are the $n-k$ disturbed columns.

Although we do not develop here, the Hidden Lattice Problem can be related to the following one, proven NP-complete by Wieschebrink in [Wi].

Punctured Code Problem

Let M be a $k \times n$ matrix, H a $k \times m$ matrix, $m \leq n$, both over a field K. Does there exist a non-singular matrix T and a subset S of $\{1..n\}$ with $|S| = m$ such that the code TM_S obtained by the deletion of the columns of S equals the code H ?

3.2 Best attack known and LLL based attacks

It is easy to show that if an attacker is able to guess the non modified columns then he is able to recover in polynomial time the special matrix associated to the wanted element. However, based on our security analysis relating the Hidden Lattice Problem to other NP-complete problem, guessing these columns has an exponential complexity in $\binom{2N}{N}$. As far as we know it is the best attack known.

The main tool for lattice based attacks is the capacity to characterize a searched solution into a vector of low weight and then to apply the LLL algorithm to find it. Although we cannot give the details here it does not seem feasible (although at first glance it may seem!). The two main reasons are following. First, it is not possible to attack only one matrix at a time since all parameters are random and hence finding an error from a random matrix without any comparison matrix cannot work. Second, if one starts from two matrices M_i and M_j, they differ at least from a random matrix $P_i^{-1}P_j$ used in their construction and hence the vectors of low weights $\{-1, 0, 1\}$ introduced for the small noise are turned into random vectors. Moreover, any attack is made even more difficult to handle since all columns are multiplied by the random elements δ_i (the diagonal terms or Δ) which scramble the vector sizes. Therefore it seems unlikely (if not impossible) that the soft noise introduced may be characterized as vectors of low weight. Of course, for the given sizes of parameter p trying to invert the action of the scrambling matrix does not seem possible as there is at least 2^{60} possibilities for each column. In our context any LLL based attack seems consequently out of not applicable to our scheme.

4 Proposed parameters and performance

4.1 Proposed set of parameters

For practical use we propose as parameters, $l_0 = 20$ ($3l_0 = 60$), and $N = 50$. The parameter $N = 50$ implies a complexity of more than 2^{100} operations to retrieve the 50 non modified columns.

Scheme	Q: d=1,d=2	E: d=1,d=2	Reply generation (b/s)
Lipmaa	2Mb, 126 Kb	2 , 3	200
Gentry and Ramzan	3Kb , 3 Kb	4 , 4	400
Ours	300 Mb, 19 Mb	6, 36	50K

Taking $l_0 = 20$ permits to make out of reach the search for non invertible submatrices of M_i (which is not in itself an attack but a first step in the direction of a potential attack). We propose to choose $p = 2^{60} + 325$. Notice that such a choice (a power of 2 plus a small integer) may also be used to fasten the computation of modular multiplication as for elliptic curves. A weak estimation of the number of elements which can be handled by such parameters is $n \leq \frac{2^{l_0}}{50} \simeq 20000$. Increasing l_0 linearly results in an exponential growth of n and therefore for any reasonable size of n this parameter does not need major changes. The expansion factor is 6.

4.2 Performance

In the following table we give peformance of our scheme for the practical case of a database of 1000 elements of size 2 MB and for a computer of 4 GHz. In the table $'Q'$ stands for the size of the query and $'E'$ for the expansion factor. The cases $d = 1$ and $d = 2$ correspond to the degree of recursivity (see [K0]).

We compare the performance of our scheme with the performance of the best known schemes of Lipmaa ([Li]) and Gentry-Ramzan ([GR]).

The results show that (as usual for lattice or code-based schemes) the size of the query for our scheme is larger than for number theory based schemes, but that eventually the reply generation speed which is the true limitation factor is a hundred times faster than for other schemes. Notice that in the context of large database, which is the standard case, the size of the query is not a so important factor overall. This makes our scheme an almost practical PIR scheme in comparison with previously known schemes which remain almost unpractical.

References

[Ga] W. Gasarch "A survey on private information retrieval" *available at http://umd.edu/ gasarch/pir/mysurvey.ps*

[GR] C. Gentry and Z. Ramzan "Single-Database Private Information Retrieval with Constant Communication Rate" *ICALP*, p.803–815, 2005.

[K0] E. Kushilevitz and R. Ostrovsky "Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval" *FOCS, 1997*

[Li] H. Lipmaa "An Oblivious Transfer Protocol with Log-Squared Communication", *ISC*, p. 314-328, 2005.

[Wi] Christian Wieschebrink "Two NP-complete Problems in Coding Theory with an Application in Code Based Cryptography", ISIT 2006, Seattle.

A McEliece-like Symmetric-Key Cryptosystem based on Quasi-Cyclic Low-Density Parity-Check Codes[1]

Ali Akbar Sobhi Afshar*, Taraneh Eghlidos**, Mohammad Reza Aref***

*Information Systems and Security Lab. (ISSL), School of Electrical Engineering,
Sharif University of Technology, Tehran, Iran.
sobhiafshar@ee.sharif.edu

**Electronics Research Center,
Sharif University of Technology, Tehran, Iran.
teghlidos@sharif.edu

***Information Systems and Security Lab. (ISSL), School of Electrical Engineering,
Sharif University of Technology, Tehran, Iran.
aref@sharif.edu

Abstract

McEliece introduced the first code-based cryptosystem that has the advantages of encryption/decryption speed and relatively easier key generation in comparison with cryptosystems based on number theory. A large public key size and a low information rate are its main disadvantages. A symmetric version of the McEliece scheme was proposed by Rao in 1986. This paper addresses the issue of using Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) codes in a symmetric-key McEliece-like cryptosystem. This method is based on a recent scheme making use of subcodes of a binary primitive BCH code, presented in CLC2006. One of the advantages of our scheme is its efficient encryption-decryption implementation. Another advantage of this scheme is the relatively smaller secret key size compared to the previous similar schemes designed to lessen the size of the key.

Keywords. McEliece scheme, Symmetric-key cryptosystem, Quasi-cyclic codes, Low-density Parity-check codes.

1 Introduction

In 1978, McEliece proposed the first public-key cryptosystem based on algebraic coding theory [Mc78]. The security of the scheme is based on the fact that the decoding problem for general linear codes is NP-complete [BM+78]. For a comparable security level, it has the advantage of high-speed encryption and decryption with respect to number-theoretic cryptosystems [PB+92]. In addition, the McEliece scheme employs probabilistic encryption [GM82], which is better than other types of deterministic encryption in preventing the partial information leaked through public-key cryptography. Despite these advantages, the McEliece public-key cryptosystem is not widely used. This is because of its large public key size and low information rate. However, some methods [LF90] have been proposed to improve the information rate of the McEliece scheme and the ever-decreasing cost of storage keeps it on the list of candidates for some applications. In the last decade, the introduction of new coding techniques has allowed to achieve near-capacity transmission over the Additive White Gaussian Noise (AWGN) channel. In particular, the codes

[1]This work was partially supported by ITRC.

invented in [Ga63], i.e., Low-Density Parity-Check (LDPC) codes, were recently rediscovered and analyzed in [Ma99]. Regular LDPC codes, introduced in [Ga63], have been extended to irregular LDPC codes, which show improved performance [CF+01]. In this paper, we discuss the efficiency of using QC-LDPC codes in a symmetric-key McEliece-like cryptosystem, and study the security-related considerations of the new scheme. These codes have encoding advantage over other types of LDPC codes. Based on the methods recently presented for encoding of QC-LDPC codes [LC+06], [CX+04], efficient implementation of our scheme is made possible. The paper is organized as follows. First, we give a brief introduction of the McEliece scheme. We then recall basic facts about quasi-cyclic codes and the construction of QC-LDPC codes based on Finite Geometry. Eventually, we introduce our new scheme and discuss its efficient encryption/decryption implementation and security.

2 THE McEliece Public-key Cryptosystem

S is a random $(k \times k)$ nonsingular matrix over $GF(2)$, called the scrambling matrix, G is a $(k \times n)$ generator matrix of a binary Goppa code G with the capability of correcting an n-bit random error vector of Hamming weight less than or equal to t, and P is a random $(n \times n)$ permutation matrix. Public key consists of (G', t). $G' = SGP$, this generates a linear code with the same rate and minimum distance as the code generated by G [Mc78].
Encryption: $c = mG' + e$, where m is a k-bit message, c is an n-bit ciphertext, and e is an n-bit random error vector of weight t.
Decryption: The receiver first calculates $c' = cP^{-1} = mSG + eP^{-1}$, where P^{-1} is the inverse of P. Because the weight of eP^{-1} is the same as the weight of e, the receiver uses the decoding algorithm of the original code G to obtain $m' = mS$. Finally, the receiver recovers m by computing $m = m'S^{-1}$, where S^{-1} is the inverse of S.

3 LDPC Codes

A linear block code C of length n is uniquely specified by either a generator matrix G or a parity-check matrix H. If it is specified by a parity-check matrix H, the code C is simply the null space of H.

Definition 3.1 *:An LDPC code is defined as the null space of a parity check matrix that has the following structural properties: 1) Rows have equal weights ρ; 2) Columns have equal weights γ; 3) No two rows (or two columns) of H have more than one 1-component in common. The constraint on rows and columns is called the row-column (RC) constraint; 4) Both ρ and γ are small compared to the length of the code and the number of rows in H.*

Because of the property (4), H is said to be a low-density parity-check matrix, and the code specified by H is hence called an LDPC code. The LDPC code given by the above definition is called a regular LDPC code. If all the columns or all the rows of the parity-check matrix H do not have the same weight, an LDPC code is then said to be irregular [Ga63], [LC04]. LDPC codes, decoded with iterative decoding based on belief propagation [Pe88], such as the sum-product algorithm (SPA) [Ma99], [RS+01], achieve amazingly good error performance.

4 Introduction to QC-LDPC Codes

Although iterative decoding of long LDPC codes can be practically implemented, encoding of these codes can be complex, since most of these codes, especially computer-generated random codes, do not have sufficient structure to allow simple encoding. One class of structured

LDPC codes that permits low-complexity encoding is the class of quasi-cyclic (QC)-LDPC codes [YL+01], [LC04].

4.1 QC Codes

DEFINITION 4.1 *: A code of length n is called quasi-cyclic of order n_0, for n a multiple of n_0, if every cyclic shift of a codeword by n_0 coordinates is again a codeword.*

A general form for the generator matrix of a (mn_0, mk_0) quasi-cyclic code with shifting constraint n_0 is:

$$G = \begin{pmatrix} G_0 & G_1 & ... & G_{m-1} \\ G_{m-1} & G_0 & ... & G_{m-2} \\ \vdots & & & \vdots \\ G_1 & G_2 & ... & G_0 \end{pmatrix},$$

where each G_i is a $k_0 \times n_0$ submatrix. As it is seen, G displays the cyclic structure among the rows and columns in terms of the submatrices G_i's. As discussed in [PW72], circulant form and quasi-cyclic form for generator matrices of QC codes are considered to be equivalent. Detailed information on quasi-cyclic codes, their construction and encoding can be found in [LC04], [PW72].

4.2 Construction of QC-LDPC codes

LDPC codes can be constructed algebraically based on the points and lines of finite geometries, such as Euclidean and projective geometries over finite fields [YL+01]. We can obtain the generator matrix of a QC-LDPC code in circulant form, whose parity-check matrix is given as an *array of circulants* [LC+06]. As discussed in section 4.1, we can obtain the generator matrix for the *designed* QC-LDPC code in quasi-cyclic form. Detailed information on the construction of QC-LDPC codes will be included in the final article.

5 New scheme based on QC-LDPC Codes

The new scheme is similar to a symmetric version of the original McEliece scheme [RN86], except that we start from the parity-check matrix of a QC-LDPC code in quasi-cyclic form, as discussed in section 4.2. We construct a quasi-cyclic parity-check matrix of a QC-LDPC code $C(n, k)$ as described earlier. We then choose a random permutation π on n_0 coordinates and construct P. This permutation matrix is in the form of a block diagonal matrix, which is a diagonal matrix in which the diagonal elements are identical permutation matrices of size $n_0 \times n_0$, *i.e.* $\pi_{n_0 \times n_0}$, and the off-diagonal elements are zero. The secret key K_s consists of the first k_0 rows of H compressed with a known lossless compression algorithm, and the permutation $\pi_{n_0 \times n_0}$.

Encryption: The sender decompresses K_s with the decompression algorithm and obtains the first k_0 rows of H, and constructs the QC generator matrix for this code as in [LC+06]. He then computes $G' = GP$, and extracts the encryption matrix G'' from G' similar to the procedure discussed in [Ga05] to obtain a unique encryption matrix with independent rows. Let m be the message to be encrypted. The ciphertext c is obtained by: $c = mG'' + eP$, where, e is an error vector obtained by first selecting a random syndrome $s \in F_2^{n-k}$ and computing $e = e(s) = s.(H^{-1})^T$. This method for choosing the error vector was first proposed by [BY98].

Decryption: The receiver has to extract the matrix G'' only once from K_s. Then he can decrypt as follow: 1)Apply P^{-1} to c. 2) Compute the syndrome for $c' = cP^{-1}$ and obtain

$s = c'H^T$. 3) Subtract the error vevtor $e = e(s) = s.(H^{-1})^T$ from c'. 4) Decode $mG''P^{-1}$ and recover m.

5.1 Performance

Quasi-cyclic codes can be encoded with shift registers with linear complexity based on their generator matrices in a simple way. Several efficient encoding implementations have been given in [LC+06]. These implementations require a complexity linearly proportional to the number of parity-check bits of the code for serial encoding, and to the code length for high-speed parallel encoding. Well-designed QC-LDPC codes have been shown to perform as well as computer-generated random LDPC codes, regular or irregular, in terms of bit-error performance, block error performance and error floor, collectively [CX+04], [LC+06]. Therefore, they are strong competitors to the random codes in practical applications due to their simple encoding and low error floors. These codes also have advantages in integrated circuit (IC) decoder implementations due to their symmetry, which results in simple regular wiring and modular structure [LC+06]. In addition to these implementation and performance advantages, we have been able to reduce the secret key to a very reasonable size. We have achieved a decrease of eight to ten times in secret key size relative to the symmetric version of the scheme in [Ga05] (which has a much smaller key size compared to [RN86]) using a lossless compression algorithm, *e.g.* the modified *Run-Length Coding*.

5.2 Security

The number of QC-LDPC codes is large by construction based on the choice of the geometry and selection of the circulants. We also have a large number of permutations based on the choice of the order of quasi-cyclicity. These two cases together with the large length of the code makes the structural attacks to recover G' infeasible. The permutation matrix P prevents the attacker to make use of the finite geometry specifications in case it is leaked. We can also increase the security of the new scheme for the same secret key size compared to a similar scheme using a general QC code [Ga05] due to the low-density structure of QC-LDPC codes and hence employ a larger code length. The speical form of the error vector with its relatively large weight makes the scheme resistant to the chosen plaintext attack in [RN86]. Typical system parameters will be discussed and compared to the parameters given in [Ga05] in the final article.

6 Conclusion

In this paper, we presented a symmetric-key McEliece-like cryptosystem based on quasi-cyclic low-density parity-check codes. Recently, efficient algorithms have been presented for the encoding of this class of codes. These implementations require a complexity linearly proportional to the number of parity bits of the code for serial encoding, and to the length of the code for high-speed parallel encoding. The QC-LDPC codes based on finite geometry also have various efficient decoding algorithms. The low-density structure of these codes makes it possible to reduce the size of the secret key to a very reasonable size based on the scheme employed and a lossless compression algorithm while retaining an acceptable level of security.

References

[AM89] C. Adams and H. Meijer, "Security-related comments regarding McEliece's public-key cryptosystem", *IEEE Transactions on Inform. Theory*, Vol. 35, No.2, pp.454-455, 1989.

[BM+78] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems", *IEEE Trans. Inform. Theory*, Vol. 24, No. 5, pp. 384-386, 1978.

[BY98] A. I. Barbero, O. Ytrehus, "Modifications of the Rao-Nam Cryptosystem", Proc. ICCC98, Guanajuato, Mexico, pp. 1-13.,1998.

[CF+01] S.-Y. Chung, J. G. D. Forney, T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit", *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58-60, Feb. 2001.

[CT06] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York:Wiley, 2006.

[CX+04] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon-limit quasicyclic low-density parity-check codes", *IEEE Trans. Commun.*, vol. 52, no. 7, pp. 10381042, Jul. 2004.

[Ga05] P. Gaborit, "Shorter keys for code based cryptography", Workshop on Codes and Cryptography, Bergen 2005, p. 81-90,and in Proceedings of CLC2006, Technische Universitt Darmstadt, Sept. 2006.

[Ga63] R. G. Gallager, *Low Density Parity Check Codes*, PhD. Thesis, Cambridge, MA, MIT Press, 1963.

[GM82] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information", in Proceedings of the 14th ACM Symposium on the Theory of Computing, pp. 270-299, 1982.

[Jo86] F. Jorissen, "A security evaluation of the public-key cipher system proposed by McEliece, used as a combined scheme", Technical Report, Dept. of Elektrotechniek, Katholieke University Leuven, 1986.

[LC04] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 2004.

[LC+06] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, "Efficient Encoding of Quasi-Cyclic Low-Density Parity-Check Codes", *IEEE Trans. Commun.*, VOL. 54, NO. 1, Jan. 2006.

[LF90] M. C. Lin and H. L. Fu, "Information rate of McEliece's public-key cryptosystem", *Electronics Letters*, Vol. 26, No. 1, pp. 16-18, 1990.

[Ma99] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices", *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 399-431, Feb. 1999.

[Mc78] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory", DSN Progress Report, 42-44, pp. 114-116, 1978.

[PB+92] B. Preneel, A. Bosselaers, R. Govaerts and J. Vandewalle, "A Software implementation of the Mceliece cryptosystem", Proceedings of the 13th Symposium on Information Theory in the Benelux, pp. 119-126, 1992.

[Pe88] J. Pearl, "Probabilistic Reasoning in Intelligent Systems", Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann, 1988.

[PW72] W. W. Peterson and E. J. Weldon, *Error Correcting Codes*, 2nd ed. Cambridge, MA: MIT Press, 1972.

[RN86] T.R.N. Rao and K.H. Nam, "A Private-Key Algebraic-Coded Cryptosystem", Proc. Crypto86, 1986, pp. 35-48.

[RS+01] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity check codes", *"IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619-637, Feb. 2001.

[YL+01] K. Yu, S. Lin, and M. Fossorier, "Low density parity check codes based on finite geometries: A discovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 11, pp. 27112736, Nov. 2001.

A Three Based Chaotic Maps Block Cipher

Mabrouka Hagui* , Hedi Khammari† and Bechir Ayeb*

* PRINCE Research Lab, FSM
Monastir

† ISATK
Kairouan

Tunisia

Abstract

The aim of this work is to look for issues about the security and realization of encryption methods to enhance the performance of the communication security. A new block chaotic cryptosystem based on three chaotic maps called TBCMBC is presented. It can provide perfect cryptographic properties. Several examples of some experiments of TBCMBC in image encryption are given.

Keywords. Chaos,Block Cipher, Image encryption

1 Introduction: Objective and Related Work

The fast evolution of technology of communication and the huge use of digital data (text, image, audio, video) in different domains increase the need of data encryption. New theories are used to enhance performance of cryptographic algorithm such as chaos. This theory was chosen for several interesting properties of the chaotic behavior. The main feature of chaos is the sensitivity to initial conditions and control parameters. In fact, two neighbor initial conditions can give rise to two chaotic orbits for the same parameter set. In comparison with conventional cryptosystems, algorithms based on chaos are easier to be realized because the chaotic maps can be merely implemented either in software or in hardware.

There are two main ways to design digital chaotic ciphers: (1) Chaotic stream cipher that produce a pseudo-random keystream which is used to mask the plaintext. (2) Chaotic block cipher that consists in using plaintext and/or the secret key(s) as the initial conditions and/or control parameters, the iterating or counter iterating chaotic system multiple times to obtain ciphertext. This paper focuses on digital chaos and essentially chaotic block encryption.

Some chaotic ciphers based on piecewise linear chaotic maps (PWLCM) have been developed in many former studies such as in [SH01],[BA89] where it has been proved some interesting statistical properties of digital PWLCM with finite computing precision. Piecewise linear chaotic map has uniform invariant density and good correlation function. These good features lead to use them to build chaotic ciphers and Pseudo-random generators. Other class of chaotic block encryption based on iteration of chaotic maps is the image encryption. this kind of algorithms uses one dimensional map such as [DSS05], [YG00] or two dimensional maps [FR98] to permutate pseudo-randomly pixels.

This paper proposes a new block of chaotic cipher that can be used for either text or image encryption. Some examples of its implementation will be presented.

2 Proposal: 3-Based Chaotic Maps

2.1 Theoretical Issues

The main PWLCM used in TBCMBC is tent map which has good dynamic properties [AS03] and is defined as follows:

$$F_p(x) = \begin{cases} x/p, & \text{if } 0 \le x \le p \\ (1-x)/(1-p), & \text{if } p < x \le 1 \\ 0 & \text{elsewhere} \end{cases} \quad (1)$$

Where p is a control parameter.

DEFINITION 2.1 *A function $G : R \to N$ is called a digital approximate transformation function , if $\forall x \in R, |G(x)-x| < 1$. The following three DATF are concerned in this paper: 1) floor(x)=$\lfloor x \rfloor$ is the integer part of x, 2) ceil(x)=$\lceil x \rceil$ =floor(x)+1, 3) round(x)=$x - floor(x)$*

For each value of control parameter p or initial condition x_0 this map generates always chaotic sequences. This PWLCM map has the desired features to be used in chaotic cryptography as mentioned in [AS03], [AS06].

This encryption approach consists in splitting the message into blocks having the same size denoted S. The size of a block is dynamic so that the parameter S is a part of the secret key and can be modified for each use of the cryptosystem. The encryption algorithm is based on three chaotic systems and can be used for either text or image encryption. To each block we associate two different sub-keys generated by the two first chaotic maps chosen here as Tent maps. The secret key is constituted by the control parameters, initial conditions of the two Tent maps, the block size S and the iterations number of the third chaotic map.

The dynamical properties of chaos allows to make the choice of key components complex, rich and somewhat tricky. The encryption scheme starts by introducing elements of secret key; then the two tent maps are iterated n times in order to generate chaotic sequences $\{x_1, x_2, ...x_n\}$ and $\{p_1, p_2, ...p_n\}$ which are respectively associated to initial conditions and control parameters. Such sequences will be employed to extract sub-keys for the block encryption . In order to encrypt a message of N plaintext blocks, the visited domains of chaotic orbit will be divided into N n_0-sized intervals, n_0=floor(n/N). Thus, the intervals X_i=$\{x_{(i-1)n_0+1}, ..., x_{in_0}\}$ and P_i=$\{p_{(i-1)n_0+1}, ..., p_{in_0}\}$, $i = 1, .., N$, are associated to the $\imath_{th}$ plaintext block. Subsequently each of these intervals is split into S sub-intervals $X_{i,j}$=$\{x_{(i-1).n_0+j.n_1+1}, ..., x_{i.n_0+(j+1).n_1}\}$ and $P_{i,j}$=$\{p_{(i-1).n_0+j.n_1+1}, ..., p_{i.n_0+(j+1).n_1-1}\}$, $_j = 0, 1, .., S-1$
with $n_1 = floor(n_0/S)$, the middle values of $X_{i,j}$ and $P_{i,j}$ denoted $X^m_{i,j}$ and $P^m_{i,j}$ are $X^m_{i,j}$= $x_{(i-1).n_0+j.n_1+n_2+1}$ and $P^m_{i,j}$= $p_{(i-1).n_0+j.n_1+n_2+1}$ where $n_2 = ceil(n_1/2)$ will be used as respectively the initial condition and the control parameter of the third chaotic map.

The third chaotic map with the parameters defined above is now used to encrypt the plaintext or plain-image.

2.2 Image Encryption

Let I be an $M \times N$ image to be encrypted. The first step is to compute sub-keys by means of two tent maps.The sub-keys correspond sequences of initial conditions and control parameters as described below. In addition to parameters indicated for the text encryption, the secret key include a new parameter called substitution key. The procedure of encryption can be dived on two stages: permutation stage and substitution stage.

The second step consists in splitting plain image into horizontal $S \times N$ sized rectangles (or blocks). Then, a PWLCM map $G_p(x)$, used as a third chaotic system, is applied in permutation stage.

$$G_p(x) = \begin{cases} x/p, & \text{if } x \leq p \\ (x-p)/(1/2-p), & \text{if } p \leq x < 1/2 \\ G_p(1-x), & \text{if } 1/2 \leq x \leq 1 \end{cases} \quad (2)$$

The variable x vary form 0 to N where N is the width of I . $G_p(x)$ is defined on the unit interval so the following adjustment is to be done on the PWLCM:

$$G_p(x) = \begin{cases} x/p, & \text{if } x \leq p \\ (x-p)/(N/2-p), & \text{if } p \leq x < N/2 \\ G_p(N-x), & \text{if } N/2 \leq x \leq N \end{cases} \quad (3)$$

Each block of the plain image comprises $S \times N$ pixels and involves an initial condition and a control parameter. The permutation procedure is executed line by line. It is based on the PWLCM which permits to interchange pixels positions. Pixels in the first column $x = 0$ keep unchanged under application of $G_p(x)$. In this case, we apply the PWLCM to the initial condition obtained from the first chaotic map in order to have the coordinates of another pixel in the same line, and permute it with the pixel with abscise $x = 0$.

In substitution stage, pixels values should be XORed with the corresponding ones in the next block. For the last block, pixels values are multiplied by the substitution key. The two stages are applied K times (K being a part of the secret key).The decryption procedure starts in the last block in order to get real values of pixels by dividing each pixel by the substitution key value . For other blocks, each pixel is XORed again with the related one in the next block. The same PWLCM is utilized to reorganize pixels. This procedure is then reiterated K times. Some improvement of the system security is obtained through adding new key elements such as the block rank NB that will be multiplied by the substitution key rather than multiplying the last one i.e for all blocks having rank less than NB, substitution process will be the same as indicated below otherwise (ranks greater than NB), each pixel will be XORed with the corresponding one on the previous block.

3 Implementation and First Experimentations

The proposed cryptosystem is implemented to encrypt firstly images. The histogram characterizes the appearance frequencies as a function of the pixel number of a given image. Analysis of histograms permits to realize the uniformity of the pixel distribution for encrypted images when compared to plain image's. An encryption software was made for this task and require the following input elements of secret key: initial conditions, control parameters, block size, iteration number and substitution key.

Note that cond1 and par1 (respectively cond2 and par2) are initial conditions and control parameters of first chaotic system (respectively second chaotic systems). For example, Figure 1 is drawn as for cond1= 0.5999, par1= 0.4562, cond2=0.8799 par2=0.112 block size=20 number iteration n=5 and substitution key=0.75. It presents some famous touristic places in Tunisia and their histograms, their associated encrypted images and their histograms.

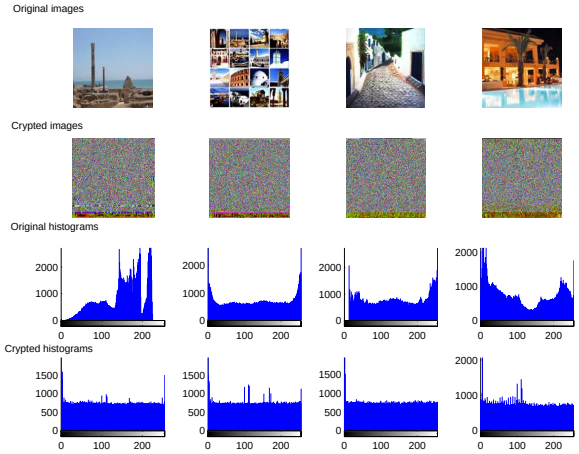


Figure 1: Experimentations Set, where cond1= 0.5999,par1= 0.4562, cond2=0.8799 par2=0.112 block size=20 number iteration n=5 and substitution key=0.75.

Chaotic systems are very sensitive to initial conditions and control parameters; in order to illustrate this property, we make small changes on one among such parameters and shows that the system behavior is completely different. Subsequently, we observe the same result when the changes affect only the control parameter.

4 Conclusion: Brief Analysis and Perspectives

Many factors can affect the performance of chaotic cryptosystem such as dynamical degradation, security and encryption speed. The proposed cryptosytem is based on three cascade chaotic maps, therefore, the number of iterations of each chaotic map can be decreased with preserving its dynamical properties. Besides, TBCMBC is implemented with double precision.

Generation of initial conditions and control parameters can be done simultaneously. Also first stages of blocks encryption (permutation of pixels) are achieved at the same time because they are independent from each other. Thus, the TBCMBC speed can be improved.

The secret key of TBCMBC is constituted by initial conditions, control parameters, block size and iteration number . Initial conditions and control parameters are given over 64 bits, whereas the block size and iteration number are written on 8 bits. For image encryption, we append substitution key (8 bits) so the secret key is represented on 280 bits detailed as: $4 * 64 + 3 * 8 = 280$. The key space is 2^{280} and the probability to obtain key is very small 2^{-280}.

In this paper, a new chaotic cipher based on cascade chaotic maps called TBCMBC was proposed and implemented in image encryption. Theoretical analysis and experiments confirm that TBCMBC has great dynamical and cryptographic properties. In further researches, TBCMBC can be improved in order to include encryption of video sequences and image compression.

References

[BA89] MS.Baptista, Cryptography with chaos, in Physics Letters A, 240:50-54.

[AS03] Gonzalo Alvarez and Shujun Li. Cryptographic requirements for chaotic secure communications, arXiv:nlin.CD/0311039 v1 20 Nov 2003.

[AS06] Gonzalo Alvarez and Shujun Li. "Some Basic Cryptographic Requirements for Chaos-Based Cryptosystems", International Journal of Bifurcation and Chaos, vol. 16, no. 8, pp. 2129-2151, 2006

[SH01] Li Shujun, Li Qi, Li Wenmin, Mou Xuanqin and Cai Yuanlong. Statistical Properties of Digital Piecewise Linear Chaotic Maps and Their Roles in Cryptography and Pseudo-Random Coding, Cryptography and coding - Proceedings of the 8th IMA International Conference (IMA-C&C'2001, December 17-19,2001, Cirencester, UK), Lecture Notes in Computer Science, vol. 2260,pp. 205-221,Springer-Verlag, Berlin, 2001.

[YG00] J.-C. Yen and J.-I. Guo. A new chaotic key-based design for image encryption and decrypion. In Proceedings of 2000 IEEE international Conference on Circuits and Systems (ISACS 2000), volume 4, pages 57-79,2004.

[DSS05] Daniel Socek, Shujun Li, Spyros S.Magliveras and Borko Furht. "Enhanced 1-D Chaotic Key-Based Algorithm for Image Encryption",in: Proceedings of the First IEEE/CreateNet International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005), pp. 406-408 (short paper), 2005

[FR98] Jiri Fridirich. Symmetric ciphers based on two-dimensional chaotic maps. Int. J. Bifurcation and Chaos, 8(6):1259-1284, 1998.

Toward a Concrete Security Proof of Courtois, Finiasz and Sendrier Signature Scheme

Léonard Dallot

GREYC, UMR 6072, Caen, France
leonard.dallot@info.unicaen.fr

Abstract

Courtois, Finiasz and Sendrier proposed in 2001 a practical code-based signature scheme. We give a reductionist security proof of a slightly modified version of the scheme.

Keywords. Cryptography, Coding theory, Signature Scheme, Concrete Security

1 Background and Definitions

Signature Schemes A *signature* σ is a bit string dependent of some secret (the secret key SK) only known by the signer and the signed message m. A public value (the public key PK) allows anyone to check the validity of the signature. A *signature scheme* S is defined by three algorithms [GMR88]:

- $\mathsf{Gen}_\mathsf{S}(1^\kappa)$, the *key generator algorithm*, is a probabilistic algorithm which, given some security parameter κ outputs a pair of a public key and a secret key (PK, SK)
- $\mathsf{Sign}_\mathsf{S}(m, SK)$, the *signing algorithm* takes as input a message m and a secret key SK and ouputs a signature σ.
- $\mathsf{Verify}_\mathsf{S}(m', \sigma', PK)$, the *verification algorithm* takes a message m', a candidate signature σ' and a public key PK as input. It outputs a bit that equals 1 if the signature is accepted and 0 otherwise. We also require that if $\sigma \leftarrow \mathrm{Sign}(m, SK)$, then $\mathsf{Verify}(m, \sigma, PK) = 1$.

We consider an *existential forgery* in a *chosen message attack* (EF-CMA). In this setting a $(\tau, q_\mathcal{H}, q_\Sigma)$-*adversary* $\mathcal{A}$ can obtain for bit string of his choice $q_\mathcal{H}$ hash values from an hash oracle $\mathcal{H}$ and q_Σ signatures from an oracle Σ. He attempts to output in time τ a valid forgery, that is a pair (m^*, σ^*) such that m^* was never requested to Σ and $\mathsf{Verify}(m^*, \sigma^*, PK) = 1$.

Definition 1.1 *The success of an $(\tau, q_\mathcal{H}, q_\Sigma)$-adversary $\mathcal{A}$ against a signature scheme S is defined by*

$$Succ_\mathsf{S}^{\mathsf{EF\text{-}CMA}}(\mathcal{A}) = \Pr_{(PK,SK)\leftarrow \mathsf{Gen}_\mathsf{S}(1^\kappa)}[\mathsf{Verify}_\mathsf{S}(m^*, \sigma^*, PK) = 1 | \mathcal{A}(PK) = (m^*, \sigma^*)]$$

A signature scheme $\mathsf{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is $(\tau, q_H, q_\Sigma, \epsilon)$-EF-CMA-secure *if for all (τ, q_H, q_Σ)-adversary $\mathcal{A}$ we have $Succ_\mathsf{S}^{\mathsf{EF\text{-}CMA}}(\mathcal{A}) \leq \epsilon$.*

Coding Theory Let $\mathbb{F}_q$ be the field with q elements. A (n, k)-*code* $\mathcal{C}$ is a linear subspace of dimension k of the vector space $\mathbb{F}_q^n$. Elements of $\mathbb{F}_q^n$ are called *words* and elements of $\mathcal{C}$ are *codewords*. A code is usually given in the form of a $(n-k) \times n$ *parity check matrix* H. The codewords of $\mathcal{C}$ are words x that satisfy $Hx^T = 0$. A *syndrome* $s \in \mathbb{F}_q^{n-k}$ is the vector $s = Hx^T$ for a word x.

We shall consider the *Goppa Parameterized Bounded Decoding problem* (GPBD) [Fin04], a variant of a well known $\mathcal{NP}$-hard problem of coding theory, the *Bounded Decoding problem*. The *Hamming weight* of a word x, denoted by $\mathsf{wt}(x)$, is the number of non-zero positions.

Definition 1.2 (Goppa Parameterized Bounded Decoding problem (GPBM) [Fin04])
Input: A $(n-k) \times n$ binary matrix H and a syndrome $s \in {\mathbb{F}_2}^{n-k}$
Ouput: A word $e \in \mathbb{F}_2^n$ such that $\mathsf{wt}(e) \leq \frac{n-k}{\log_2 n}$ and $He^T = s$

2 Our contribution

Our main goal is to give a rigorous security analysis of the signature scheme proposed in [CFS01]. In the original scheme, messages are hashed together with an incremented index until a correctable syndrome is found. Then if a message m is signed to a pair (i, σ), an adversary knows that for all indexes $j \leq i$, $h(m\|i)$ is not correctable. To avoid these situation, we propose to slightly modify the scheme in the following way:

- $\mathsf{Gen}_{\mathsf{mCFS}}(n, k)$: randomly choose a (n, k)-Goppa binary code $\mathcal{C}_0$ correcting t errors and denotes H_0 its parity check matrix. Choose a random $(n-k)\times(n-k)$ non-singular matrix U, a random $n \times n$ permutation matrix P and a hash function $h : \{0,1\}^* \longrightarrow \mathbb{F}_2^{n-k}$. Then, compute $H = UH_0P$ and output H_0, U and P as the secret key, H as the public key and h and $t = \frac{n-k}{\log_2 n}$ as public parameters.
- $\mathsf{Sign}_{\mathsf{mCFS}}(m, H_0)$: choose a random integer $i \in_R \{1, \ldots, 2^{n-k}\}$. Compute $s = h(m\|i)$ and decode $U^{-1}s$ using H_0 into a word x' of weight $\mathsf{wt}(x') \leq t$. If an x' is found compute $x = x'P$, the pair (x, i) is the signature of m. Otherwise set i to an other random value and repeat the hashing and decoding operations until a correctable syndrome $s = h(m\|i)$ is found.
- $\mathsf{Verify}_{\mathsf{mCFS}}(m, x', i', H)$: compute $s' = Hx'^T$ and $s = h(m\|i)$. The signature is valid if s and s' are equals.

Remark. The decoding operation in the signing algorithm Sign succeeds with probability $1/t!$ due to the properties of Goppa codes [CFS01].

With this modification on the original scheme, we state the following:

Theorem 2.1 *Suppose there exists a (τ, q_H, q_Σ)-adversary who breaks the scheme with probability ϵ, then there exists an algorithm in the Random Oracle Model that solve GPBD in time τ' with probability at least ϵ' where*

$$\tau' = \tau + (q_H + q_\sigma + 1)\mathsf{T}_s \quad \textit{and} \quad \epsilon' = \frac{\epsilon}{q_\sigma + q_H + 1} + O(\tfrac{1}{2^{n-k}})$$

and T_s represent the computation time of a syndrome (wich is $0(n^2)$).

3 Conclusion and Future Works

We have proven the security of a slightly modified code-based signature scheme proposed by Courtois, Finiasz and Sendrier. Our next step is to prove the optimality of this proof (as proposed in [DR02]) in order to settle the exact security of the scheme.

References

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In *Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer, 2001.

[DR02] Yevgeniy Dodis and Leonid Reyzin. On the power of claw-free permutations. In *Security in Communication Networks: Third International Conference, SNC 02*, volume 2576 of *Lecture Notes in Computer Science*, pages 55–73. Springer, 2002.

[Fin04] Matthieu Finiasz. *Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique.* PhD thesis, INRIA – Ecole Polytechnique, October 2004.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ron L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

HFE - solved or saved?

Jintai Ding*, Dieter Schmidt** and Fabian Werner†

* University of Cincinnati ding@math.uc.edu
** University of Cincinnati dieter.schmidt@uc.edu
† Technical University of Darmstadt fw@cccmz.de

Abstract

In this paper, we will analyze the complexity of the algebraic attack on HFE (Hidden Field Equation) cryptosystems in different finite fields, and check our new claims using the Faugère F_4 algorithm in Magma. We will also present a new variation of an HFE system that can resist the existing attacks.

Keywords. HFE, Gröbner basis, multivariate public key cryptosystem

1 The HFE system

A family of multivariate public key cryptosystems was suggested by Matsumoto and Imai in 1988 [MI88]. Their basic idea was to take a small finite field k - having q elements - a level n algebraic extension called $K \cong k^n$ and the isomorphism defined as

$$\phi : K \to k^n, \quad \phi(a_0 + a_1 t + ... + a_{n-1}t^{n-1}) \to (a_0, a_1, ..., a_n).$$

The key trapdoor is a polynomial in K, usually called $\tilde{F}(X)$. The structure of $\tilde{F}(X)$ depends on the actual cryptosystem. The keypair is constructed in the following way: first, two affine linear transformations $L_1, L_2 : k^n \to k^n$ are being chosen. Next, the following is estimated for general x_i: $F(x_1, ..., x_n) = (L_1 \circ \phi \circ \tilde{F} \circ \phi^{-1} \circ L_2)(x_1, ..., x_n)$. This results in n at most quadratic polynomials over k called $f_1, ..., f_n$, each in n variables. In 1996, Jacques Patarin presented the classical HFE cryptosystem where $\tilde{F}(X)$ is defined as:

$$\tilde{F}(X) = \sum_{i=0}^{r_2}\sum_{j=0}^{i} \alpha_{ij} X^{q^i+q^j} + \sum_{i=0}^{r_1} \beta_i X^{q^i} + \gamma$$

where $\alpha_{ij}, \beta_i, \gamma$ are randomly chosen from K and r_1, r_2 not "too big" (usually ≤ 6). Encryption of a plaintext p is simply done by evaluating the public key polynomials $(c_1, ..., c_n) = (f_1(p), ..., f_n(p))$. Decryption is done by inverting all the operations in the respective order. For an attacker, the way to attack directly is called the algebraic attack. This consists of solving a set of polynomial equations, which in general is $\mathcal{NP}$-hard [GaJ79].

2 Cryptoanalysis of HFE

In 1999, Faugère presented an improved Gröbner bases algorithm called F_4 [Fa02], which was used in the algebraic attack on HFE, namely to solve the polynomial systems HFE produces:

$$f_1(x_1, ..., x_n) = c_1, ..., f_n(x_1, ..., x_n) = c_n.$$

As the ground field k for HFE was suggested to be GF(2), the so-called field equations were easily used. Field equations are of the form $x_i^q \equiv x_i$ and contain the necessary information that the algorithm should figure out all solutions to the system of equations over k not over the extension field of k. Our key point is that when we change the characteristic of the ground field to something like 11, things look different. We have two different approaches to solve the system now: inserting the field equations and leaving them out. We will argue that the complexity of the algebraic attack should be exponential.

2.1 Leaving out the field equations

When omitting the field equations, the Gröbner basis algorithm estimates the variety of the system of polynomials but not over K. Instead, the algorithm calculates the whole variety in the algebraic closure of K. From the Fundamental theorem of algbera, we know that this system has the same amount of solutions (denoted by D) as the degree of $\tilde{F}(X)-p'$, where $p'=\phi^{-1}(p)$ and p is a plaintext. Statistically speaking, we know that choosing a random polynomial in K, we have a very high propability that it is irreducible. Therefor, it is very likely that $\tilde{F}(X)-p'$ really has D solutions. When calculating a Gröbner basis $G=\{g_1(x_1,...,x_n),...,g_n(x_n)\}$, we also have a high propability for $g_1,...,g_{n-1}$ having a degree 1. This means that $g_n(x_n)$ will have degree D. From this, it follows that the Gröbner basis for solving this system has to deal with polynomials with a degree of D. When setting $k=GF(11), r_1=r_2=2$, D will be $11^2+11^2=164$. Consequently we can not succeed in the algebraic attack for $n\geq 64$.

2.2 Inserting the field equations

When we insert the field equations for $k=GF(11), r_1=r_2=2$, the Gröbner basis algorithm should proceed in the way that it avoids the usage of the field equation until it deals with polynomials of degree 11, because this is the first point, where the field equations can actually be utilized. Depending on n, these polynomials will have $\frac{(n+d)!}{n!*d!}$ possible monomials which already is bigger than 2^{32} for $n=32$. This means that for $n\geq 64$, we don't stand a chance to deal with the polynomials of the Gröbner basis calculation.

3 Conclusion

Our claim is that in the case of using a base field with high characteristic, the algebraic attack on the HFE cryprosystems should be expoenetial in the range we consider practical. We confirm our claims with extensive computer experiments (see Figure 1) in Magma. We also build a new secure HFE variant which could resisit the algebraic attack and all other existing attacks.

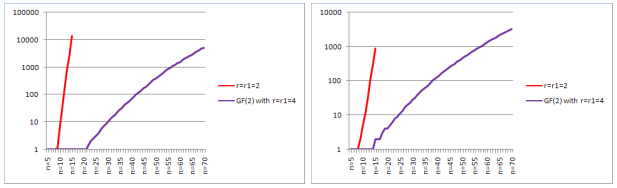


Figure 1: Timings and memory usage for HFE systems over *GF(11)*

References

[GaJ79] M.Garey and D. Johnson: *Computers and Intractability – A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979. ISBN 0-7167-1044-7.

[MI88] Tsutomu Matsumoto, Hideki Imai: *Public quadratic polynomial-tuples for efficient signature verification and message-encryption.* (1988) In EUROCRYPT 1988, volume 330 of LNCS, pages 419-545.

[Fa02] Jean-Charles Faugère : *A new efficient algorithm for computing Gröbner bases (F_4)* (2002). Université Paris. `http://fgbrs.lip6.fr/@papers/F99a.pdf`

AES Cache-timing attacks verified

Joris Plessers[1], Ingrid Verbauwhede

KULeuven - Dept. ESAT/COSIC
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
`joris.plessers@esat.kuleuven.be`

Abstract

In this paper, we will present our evaluation of two different cache-attacks on the AES-algorithm, namely the ones from Bernstein [1] and Bonneau [2]. We give the results of the reproduction of these attacks. The attack of Bernstein works on the first round and uses cache misses to perform the attack. It needs about $2^{27.5}$ samples. The attack of Bonneau works on the last round and uses cache hits to crack the key. It can be performed with less than 2^{15} samples.

Keywords. cache-timing attacks, side-channel attacks, AES

1 Introduction

Since the last decade there has been a lot of research on side-channel attacks, besides cryptanalysis. A cipher can be seen as a mathematical function, $E_K[P]=(C)$. By implementing this cipher, the mathematical function changes into $E_K[P]=(C,t)$, and some side-channels t are introduced. These can be used to break the secret key.

Modern processors have cache memory, which is much faster than main memory. Cache memory is introduced to bridge the gap between the slow main memory and the fast processor. The difference in time for accessing both memories can be used as a side-channel. We present our evaluation and comparison of two papers that show how we can use a timing attack in practice. We will discuss the results of the reproduction of both attacks. They both attack the OpenSSL implementation of AES [3]. To gather precise timings, they read out the processor's cycle counter. The first paper [1], published by Bernstein, attacks the first round of the AES-algorithm, while the second paper [2], published by Bonneau, attacks the last round.

Most software implementations, including OpenSSL, combine the AES $SubBytes$, $ShiftRows$ and $MixColumns$ operation in s-boxes. By combining these operations, one round exists of 16 s-box lookups in 4 different s-boxes and 16 xor operations. The last round omits the $MixColumns$ operation, as it would not lead to extra security and would cost extra processing time in hardware. Therefore, the 16 lookups in the last round, are performed in only one s-box.

When an element is looked up twice in the same s-box, the element will be cached and therefore it will be loaded faster. This time difference can be used as a time side-channel.

2 First round attack

The attack described by Bernstein [1] first captures a profile of the cache behaviour with a known key by making use of high timings introduced by cache misses. Then it will capture differences in timing, introduced by encryption with the unknown key. By correlating the data

[1]This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by FWO G.0300/700

of the unknown key with the captured data of the known key, we can limit the amount of possible keys. After this correlation the key space is small enough to perform a brute force attack at the key, in which each key will be checked against a known plaintext/ciphertext pair. Bernstein states that for this attacks he needs about $2^{27.5}$ samples.

We tried to reproduce this attack on two different CPU's. The first attack was done on an AMD K7 1000 MHz. On this CPU, we could only break part of the key. The second server we attacked was an AMD Athlon 2000+. On this server, we were able to break the key once, all other attempts could only brake the key partially. The difficulties we encounter, might be due to noise, introduced by programmes running in the background. By terminating unnecessary programmes in the background, the noise can be reduced.

3 Last round attack

Bonneau describes in his paper [2] an attack on the last round of the AES-algorithm. This attack will generate a large number of timing samples, produced by doing one encryption, each time followed by a programme that evicts the cache. For breaking the key it will look at the minimum time needed for lookups, which means it will use cache hits.

To break the key, we take some samples and look at the minimum timing needed for encryption. Out of these timings, the value of the key bytes can be guessed. This guess will be used to check the key at a known plaintext/ciphertext pair. If the key cannot be guessed with these samples, more samples will be used to guess and check the key, until the key is found. Bonneau states that about 2^{15} samples are needed to crack the key.

The reproduction of this attack was successful. We tried this attack on the AMD Athlon 2000+ and the AMD K7 1000 MHz and the attack gave good results on both CPU's. For the AMD Athlon 2000+ we tried to crack ten different keys, hundred times each. These attacks needed at most less than 2^{15} samples which takes less than one minute on this CPU. These results correspond to those mentioned in the paper.

4 Conclusion

We took a thorough look at two different cache-attacks, and tried to reproduce them. The attack at the first round needs a lot of samples, and thus takes a long time to perform. Besides this, we were unable to reproduce the attack succesfully, probably due to background noise. The attack on the last round of AES was more succesfull and can be performed in less than one minute. This attack is less sensitive to noise and can be a good starting point to do further research on cache attacks on embedded systems.

Both attacks make use of the cycle counter inside the CPU, which gives very precise timing. However this might not seem a threat in real life, further research needs to be done on attacks which use less precise timing, like the clock of a computer.

References

[1] Daniel J. Bernstein, *Cache-timing attacks on AES*, 2004, `http://cr.yp.to/papers.html#cachetiming`.

[2] Joseph Bonneau and Ilya Mironov, *Cache-collision timing attacks against AES*, 2006, In volume 4249 of Lecture Notes in Computer Science, pages 201-215, Springer.

[3] OpenSSL, *OpenSSL: The open source toolkit for SSL/TLS*, `http://www.openssl.org/`

A Lightweight Hardware Implementation of the Stream Cipher VEST-4

Timo Gendrullis, Timo Kasper, and Christof Paar

Chair for Communication Security, Ruhr-Universität Bochum
`http://www.crypto.rub.de`
`timo.gendrullis@rub.de,tkasper@crypto.rub.de,cpaar@crypto.rub.de`

Abstract

The use of stream ciphers for cryptographic devices in restricted environments like RFID (Radio Frequency Identification) is an emerging area. The need for secure communication demands for lightweight implementations of ciphers which meet the respective low power and small area requirements. For this purpose a lightweight version of the hardware-dedicated stream cipher VEST-4 was implemented in Hardware.

Keywords. Stream Cipher, Hardware Implementation, ASIC, VHDL

1 A Lightweight Version of VEST-4

Besides the control logic the stream cipher VEST-4 consists of a counter, a counter diffusor, a core accumulator, and an output combiner as described in [SON06]. The residue number system (RNS) counter provides three 11-bit wide and 13 10-bit wide bijective non-linear feedback shift registers (NLFSR). These NLFSRs with their coprime period length are implemented as LFSRs updated by nonlinear feedback functions. The counter diffusor then compresses some selected RNS counter output bits with a set of 6-bit wide feedback linear combiners into 10 bits. The 83-bit wide core accumulator consists similarly to the RNS counter of bijective NLFSRs but with parallel feedback (NLPFSRs). This is achieved by updating each bit with a different nonlinear Boolean function with a different set of input bits previously stored in the stream. The counter diffusor bits and the feedback of the four ciphertext bits in the implemented Authenticated Encryption (AE) mode are accepted as input bits. The 163 bits of the RNS counter, the 10 bits of the counter diffusor, and the 83 bits of the core accumulator form a 256-bit wide internal state. As the last entity the output combiner compresses 24 bits of the core accumulator with 6-bit wide linear combiners to four output bits. After the initialization phase VEST-4 accepts 4 bits of plaintext input per clock cycle and produces 4 bits of ciphertext output per clock cycle.

In opposite to the full-featured version of VEST-4 the lightweight version does not support family keying with the possibility to generate other independent cipher families. It only uses the root family instead. As another method to simplify the cipher, the 256-bit wide internal state is initially loaded directly into the registers using multiplexers and an 8-bit wide input. This is considerably faster than loading a 160-bit key using the VEST keying-mode. Additionally, it reduces the number of states of the FSM controlling the cipher.

2 The VHDL Implementation

The lightweight version of VEST-4 was implemented in VHDL and was synthesized for the AMI Semiconductor standard cell library based on the MTC45000 $0.35\,\mu m$ process which is approriate for RFID devices. The values for this process are 2.7 Volt for the core voltage and 25° C for the temperature. The tool used for simulation was *Mentor Graphics ModelsimXE III 6.0a* and the tool used for synthesis and power simulation was *Synopsis Design Compiler Version Y2006.06*.

The RFID constraints concerning the maximal mean current consumption of $15\,\mu A$ and the response time of $300\,\mu s$ should be met. Therefore, the registers in our lightweight implementation are clock gated, although they are not serialized. A serialization of the registers would have improved the power consumption on the one hand but would have enlarged and decelerated the cipher on the other hand. Nevertheless, clock-gating without register serialization reduces both, area and power consumption. To be able to compare the synthesis results to other stream cipher implementations, it was necessary to match some design definitions like clock-synchronous inputs.

In Table 1 the results of low-power implementations of the ciphers *Trivium*, *Grain*, and *AES-128* by M. Feldhofer are compared to a full-featured VEST-4 implementation by Sean O'Neill and the presented lightweight implementation. Therefore, the security of the cipher in bits, the duration in clock cycles to encrypt a 128-bit message, the area in gate equivalents and the power dissipation at a clock frequency of 100 kHz are listed. The duration to encrypt a message is divided into the amount of clock cycles needed to initialize the cipher with the key and an initialization vector and the amount of clock cycles needed to encrypt the 128 bits.

Table 1: Comparison of Different Stream Cipher ASIC Implementations

Algorithm	**Sec.** [*bits*]	**Encrypt 128bits** [*cycles*]	**Area** [GE]	**Power @100kHz** [μW]	**Properties**
Grain	80	130 + 104	3360	1.20	by M. Feldhofer:
Trivium	80	1603 + 176	3090	1.02	low-power impl.
AES-128	128	1016 + 16	3400	4.50	$0.35\mu m, 1.5V$, *w/o MAC*
VEST-4 ff	80	–	5392	–	by S. O'Neill: full-featured
VEST-4 lw	80	76 + 108	3407	1.16	$0.35\mu m, 2.7V$, *w/ MAC*

3 Conclusion

Although the lightweight implementation of VEST-4 supports authenticated encryption and a message authentication code (MAC) it can compete with low-power implementations without these features or even outperform them.

References

[SON06] Sean O'Neil, Benjamin Gittins, Howard A. Landman. VEST Ciphers. `http://www.vestciphers.com`, August 2006.

[FEL07] Martin Feldhofer. Comparison of Low-Power Implementations of Trivium and Grain. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/027 (SASC 2007). `http://www.ecrypt.eu.org/stream`, February 2007.

Cryptanalysis of the MOR System (extended abstract)

Anja Korsten

Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
`http://www-dm.informatik.uni-tuebingen.de`
`akorsten@informatik.uni-tuebingen.de`

Abstract

The MOR cryptosystem was introduced in 2001 as a new public key cryptosystem based on non-abelian groups. This paper describes a security analysis of MOR on the group $GL(n,q)\times_\theta H$.

Keywords. Public key cryptography, Discrete logarithm problem, Non-abelian group, MOR cryptosystem, Inner automorphism problem

1 Introduction

In recent years there has been considerable interest in public key cryptosystems based on non-abelian groups, *e.g.* braid groups and linear groups. In 2001 S. H. Paeng et al. introduced the MOR cryptosystem, see [2]. This ElGamal-type public key cryptosystem is based on the intractability of the *discrete logarithm problem* (DLP: Given a finite group G and $g \in G$, determine $a \in \mathbb{N}$, $0 \leq a < ord(g)$ from $h = g^a$). Paeng et al. use the fact that there is no subexponential-time algorithm known to solve the discrete logarithm problem in the inner automorphism group of non-abelian groups. The authors argue that even if the discrete logarithm problem in the underlying group G is efficiently solvable their cryptosystem is applicable to G. This paper describes an attack on the MOR system with groups of type $GL(n,q)\times_\theta H$ for which previously known attacks [3, 4, 5] do not work.

2 The MOR cryptosystem

Consider the inner automorphism group $Inn(G)$ of a non-abelian finitely generated group G in which the representation problem is efficiently solvable. In the setting of the MOR cryptosystem the inner automorphism I_g induced by an element $g \in G$ will be represented as the set of the values of I_g on the generators of G. With this representation it is possible to calculate the value of I_g on any element of G without knowing g. The MOR scheme is described as follows:

Key generation: Bob chooses an arbitrary element $g \in G\backslash Z(G)$: the private key is $a \mod ord(I_g)$, the public key is $(I_g,\ I_{g^a})$.

Encryption: Alice takes a plaintext $m \in G$, chooses an arbitrary private encryption exponent $b \mod ord(I_g)$ and computes $E := I_{g^a}^b(m)$ and $\varphi := (I_g)^b$ and sends $(E,\ \varphi)$ to Bob.

Decryption: Bob receives a ciphertext $(E,\ \varphi)$ and computes $\varphi^{-a}(E) = m$.

3 MOR on $GL(n,q)\times_\theta H$

In [2] the semidirect product $SL(2,p)\times_\theta \mathbb{Z}_p$, where θ is a certain homomorphism from $\mathbb{Z}_p$ into $Inn(SL(2,p))$, is proposed as a group to be used in the MOR scheme. On this group MOR has already been analysed and found insecure by C. Tobias and Paeng et al. in [3, 4, 5]. The techniques presented in their works are based on two facts:

- The special type of the homomorphism θ allows an immediate reduction of MOR on $GL(2,p) \times_\theta \mathbb{Z}_p$ to MOR on $GL(2,p)$ (and likewise for MOR on $SL(2,p) \times_\theta \mathbb{Z}_p$).
- In $GL(2,p)$, one can compute directly two of the four entries in the plaintext matrix from the ciphertext, which is a conjugate of the plaintext. The additional (and efficient) computation of an element of the centralizer of the enciphering element allows an adversary to obtain a third entry of the plaintext matrix.

One can avoid this type of attack by taking $n \times n$-matrices instead of 2×2-matrices and by using different homomorphisms θ. Thus we analyse MOR on the group $GL(n,q) \times_\theta H$ where $q = p^m$ is a prime power, $n \in \mathbb{N}$, H is any abelian group, and θ is a homomorphism of H into $Aut(GL(n,q))$ such that $\theta(H)$ is in one of the four basic subgroups of $Aut(GL(n,q))$.

Assuming that the discrete logarithm problem in small extension fields of $\mathbb{F}_q$ is efficiently solvable, an attack is developed to which MOR succumbs in the new setting. This consists of three steps:

1. The *inner automorphism problem* (IAP: Given I_g as a set of values on generators, determine $h \in G$ such that $I_h = I_g$) is efficiently solvable in $GL(n,q)$. It follows that the discrete logarithm problem in $Inn(GL(n,q))$ is reduced to the the *generalized discrete logarithm problem* (gDLP: Given a finite group G and $g \in G$, determine $a \in \mathbb{N},\ 0 \le a < ord(g)$ and $z \in Z(G)$ from $h = zg^a$) in $GL(n,q)$.
2. Construction of an algorithm based on work of Menezes and Wu (see [1]) which reduces the gDLP in $GL(n,q)$ to the DLP in small extensions of $\mathbb{F}_q$ (*i.e.* $\mathbb{F}_{q^d}\ d \le n$) in probabilistic polynomial time.
3. For each of the four typical subgroups of $Aut(GL(n,q))$ we are able to break MOR on the group $GL(n,q) \times_\theta H$.

Given only the public encryption key, a combination of these three parts enable an adversary, who is able to compute discrete logarithms in small extension fields of $\mathbb{F}_q$, to solve the DLP in $Inn(GL(n,q) \times_\theta H)$ and thus determine an equivalent of the secret encryption key. The MOR cryptosystem in this setting is therefore at most as secure as an DLP in $\mathbb{F}_{q^d},\ d \le n$.

Remark: The first part of the described attack can even be generalized. If the IAP in a group G is efficiently solvable, the DLP in $Inn(G)$ is polynomially reducable to the gDLP in G. Furthermore, if the order of the center of G and of $G \setminus Z(G)$ are coprime, it is even reducable to the DLP in G. This implies that in certain cases it is indeed important for the security of MOR whether the discrete logarithm problem in the underlying group is efficiently solvable.

Conclusion: In light of the presented attack it seems reasonable to consider other types of groups (instead of semidirect products of full linear groups with abelian groups) and investigate the security of the corresponding MOR systems.

References

[1] A. J. Menezes, Y. Wu, *The discrete logarithm problem in GL(n,p)*, Ars Combinatoria, Volume 47, 1998, pp. 23 - 32.

[2] S. H. Paeng, K. D. Ha et al., *New public key cryptosystem using finite non abelian groups*, CRYPTO 2001, 2001, LNCS 2139, pp. 470 - 485.

[3] S. H. Paeng, D. Kwon et al., *Improved public key cryptosystem using finite non abelian groups*, Cryptology ePrint Archive, http://eprint.iacr.org/2001/066, 2001.

[4] C. Tobias, *Security analysis of the MOR cryptosystem*, Proceedings of the PKC 2003, 2003, LNCS 2567, pp. 175 - 186.

[5] C. Tobias, *Design und Analyse kryptographischer Bausteine auf nicht-abelschen Gruppen*, PhD thesis, University of Giessen, 2004.

On the Permutations of KHAZAD

Ralph Wernsdorf

Rohde & Schwarz SIT GmbH,
Am Studio 3, D-12489 Berlin, Germany
http://www.rohde-schwarz.com
ralph.wernsdorf@rohde-schwarz.com

Abstract

It is shown that the round functions of the block cipher KHAZAD generate the alternating group on $\mathrm{GF}(2^8)^8$.

Keywords. KHAZAD, block cipher, permutation, alternating group

1 Introduction

KHAZAD [1] is a block cipher with block length 64 bit, submitted in 2000 to the NESSIE project [6] by V. Rijmen and P. Barreto and updated in 2001. KHAZAD was amongst the final candidates of NESSIE. No security flaws were found, but "... concerns were expressed with regard to the structural symmetry of KHAZAD" (see NESSIE final documents [6]). There are many similarities to Rijndael [3] but the 8×8–S–box is built out of two 4×4–S–boxes. In publications of Biryukov [2] and Muller [5] theoretical attacks on KHAZAD reduced to 5 rounds are described. Implementations of KHAZAD exist for example in the Linux Kernel (*www.linuxhq.com/kernel/v2.6/*), in the GNU Crypto project (*www.gnu.org/software/gnu-crypto/manual/Ciphers.html/*), and in the Perl Network CPAN (*search.cpan.org/~jcduque/Crypt-Khazad-1.0.3*). Because the components of the round functions are involutions the question arises which permutation group these round functions can generate. If the generated group is too small then this points to cryptographic weaknesses. Some results similar to [4] can be shown for the KHAZAD round function: Invariants, short cycles, complementation-like properties. This provides a further motivation for the determination of the permutation group generated by the KHAZAD round functions.

2 Descriptions and Notations

We will represent the field $\mathrm{GF}(2^8)$ as $\mathrm{GF}(2)[x]/p_8(x)$, where $p_8(x) = x^8 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$. The round functions of KHAZAD are defined by the concatenation of the following three permutations (for details see [1]): The nonlinear layer $\gamma\colon\ \mathrm{GF}(2^8)^8 \to \mathrm{GF}(2^8)^8$ consists of the parallel application of a nonlinear substitution box $S\colon\ \mathrm{GF}(2^8) \to \mathrm{GF}(2^8)$ to all bytes of the argument individually. The linear diffusion layer $\theta\colon\ \mathrm{GF}(2^8)^8 \to \mathrm{GF}(2^8)^8$ is a linear mapping $\theta(a) = a \cdot H$, where H is a symmetric and unitary matrix. The affine key addition $\sigma(k)\colon\ \mathrm{GF}(2^8)^8 \to \mathrm{GF}(2^8)^8$ consists of the bitwise addition of a key vector $k \in \mathrm{GF}(2^8)^8$.

3 Properties of the S-Box and of the Diffusion Matrix

For the proof in the next chapter we need:

Lemma 3.1 *The $2 \cdot 256 - 1$ permutations $x \mapsto a \oplus x$ and $x \mapsto S^{-1}(a \oplus S(x))$ $(x, a \in \mathrm{GF}(2^8))$ generate the alternating group on $\mathrm{GF}(2^8)$.*

This can be shown by computing cycles for combinations of these permutations (see [7]). For the proof in Chapter 4 we also need:

Lemma 3.2 *There is a vector* $b \in \mathrm{GF}(2^8)^8$ *such that for all non-zero diagonal byte matrices* $D = (d_{i,j})_{i=0,\ldots,7,\ j=0,\ldots,7}$, $d_{i,j} \in \{0,1\}$ *every component of the vector* $b \cdot D \cdot H \in \mathrm{GF}(2^8)^8$ *is non-zero.*

By random search we found several such vectors b, for example $b =$ 76 08 05 3f 4f 4e f2 74.

4 The Generated Group is the Alternating Group

Similar to [7] the following steps lead to a proof that the round functions of Khazad generate the alternating group on $\mathrm{GF}(2^8)^8$:

a) The generated group is transitive (similar to [7]).

b) The generated group contains only even permutations (similar to [7]).

c) Every parallel application of 8 arbitrary even byte-permutations belongs to the generated group (Application of Lemma 3.1).

d) The subgroup generated by those group elements that let the all-zero-vector fixed is transitive on $\mathrm{GF}(2^8)^8 \setminus \{(0,0,\ldots,0)\}$ (Application of Lemma 3.1 and Lemma 3.2). Together with a) this implies that the generated group is 2-transitive.

e) In the generated group there exists a permutation different from the identity permutation that lets $2^{64} - 3 \cdot 2^{56}$ elements fixed (Application of c) and Lemma 3.1).

f) Finally by Theorem 15.1 in [8], from b), d) and e) it follows that the round functions generate the alternating group.

5 Conclusions

The round functions of Khazad generate the alternating group on $\mathrm{GF}(2^8)^8$. This shows that the special structure of the round function components of Khazad has no consequences with respect to the size of the generated group.

References

[1] P. Barreto and V. Rijmen. The Khazad Legacy–Level Block Cipher. *http://paginas.terra.com.br/informatica/paulobarreto/KhazadPage.html*

[2] A. Biryukov. Analysis of involutional ciphers: Khazad and Anubis. In *Fast Software Encryption 2003*, Lecture Notes in Computer Science 2887, pp. 45–53, Springer-Verlag, 2003.

[3] J. Daemen and V. Rijmen. AES Proposal: Rijndael. NIST AES Proposal, 1998.

[4] T. Van Le, R. Sparr, R. Wernsdorf, and Y. Desmedt. Complementation-like and cyclic properties of AES round functions. In *Advanced Encryption Standard - AES, 4th International Conference, AES 2004*, Lecture Notes in Computer Science 3373, pp. 128–141, Springer-Verlag, 2005.

[5] F. Muller. A new attack against Khazad. In *Advances in Cryptology - ASIACRYPT 2003*, Lecture Notes in Computer Science 2894, pp. 347–358, Springer-Verlag, 2003.

[6] NESSIE, *New European Schemes for Signature, Integrity, and Encryption.* `http://cryptonessie.org`, 2000.

[7] R. Wernsdorf. The round functions of Rijndael generate the alternating group. In *Fast Software Encryption, FSE 2002*, Lecture Notes in Computer Science 2365, pp. 143–148, Springer-Verlag, 2002.

[8] H. Wielandt. *Finite Permutation Groups.* Academic Press, New York, 1964.

Exploring Trade-offs between Area, Performance and Security in HW/SW Co-design of ECC [1]

Caroline Vanderheyden, Junfeng Fan, Kazuo Sakiyama, and Ingrid Verbauwhede

Katholieke Universiteit Leuven,ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{cvanderh,jfan,ksakiyam,iverbauw}@esat.kuleuven.be

Abstract

Elliptic Curve Cryptography (ECC) is increasingly used in cryptographic applications such as the Diffie-Hellman key agreement and the Digital Signature Algorithm. The main operation in ECC is scalar multiplication, which is performed by repeating point additions and doublings. Every point addition and doubling consists of several modular multiplications and additions. In this paper we implement an ECC coprocessor over GF(2^m). Four different architectures are investigated to achieve a trade-off between area, performance and security. Countermeasures for power analysis attacks are explored in both circuit level and algorithm level.

1 System Architecture: MicroBlaze with coprocessor

As starting point, we use a Xilinx MicroBlaze softcore and a coprocessor, called Modular Arithmetic Logic Unit (MALU) [1]. The MALU can perform modular multiplications and additions. We propose four types of architectures to explore the trade-off between area, performance and security (Figure 1). In the first architecture, Type-A, the MALU and a Register File (RF) are used in the coprocessor which is controlled by the MicroBlaze processor. The MicroBlaze sends 32-bit instructions including `Store`, `Load`, `ModMul` and `ModAdd` in order to perform point additions and doublings. In the Type-B architecture, we allocate the instructions into a micro-code ROM. This reduces the communication overhead between the MicroBlaze and the coprocessor. In the Type-C architecture, we assign a register into the coprocessor to store the key to accelerate the scalar multiplication. In Type-D, the key is randomized to achieve Differential Power Analysis (DPA) resistance by adding a Random Number Generator (RNG).

2 Countermeasures for Side-channel Attacks

Resistance against side-channel attacks is now an important dimension in the whole design. By deploying the Montgomery scalar multiplication, ECC can be secured against Timing Analysis (TA) and Simple Power Analysis (SPA) attacks. To secure ECC against DPA, there are several possible countermeasures such as Wave Dynamic Differential Logic (WDDL) [2] on circuit level, and key randomization, an algorithm level countermeasure proposed in [3]. Let $\#E$ be the number of points of the curve, we have $\#E \cdot P = \mathcal{O}$. Therefore the private key, d, can be randomized to avoid DPA. We have $d' \cdot P = d \cdot P + k \cdot \#E \cdot P = (d + k \cdot \#E)P$, here k is a random number of size n bits. This countermeasure makes ECC resistant against DPA, since the new key d' in $Q = d'P$ changes every time the calculation of this is launched.

[1] This paper was supported in part by the European Commission under the ECRYPT Network of Excellence in Cryptology (G.0450.04, G.0475.05). This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by the EU IST FP6 projects (SESOC and ECRYPT), and by the IBBT-QoE project of the IBBT.

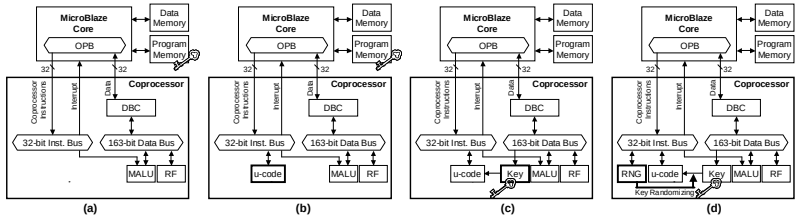


Figure 1: Various hardware options for the ECC coprocessor; (a) Type-A: Baseline coprocessor implementation with the MicroBlaze. (b) Type-B: micro-code ROM is added for accelerating point addition/doubling. (c) Type-C: key register is added for accelerating scalar multiplication. (d) Type-D: RNG is added for randomizing key.

Table 1: Area, performance and security trade-offs for four different architectures.

Architecture	**Area** [slices]	**Prog.ROM** [bits]	**Perform.** [Kcyc.]	**Comments**
Type-A	4307	-	1254	TA/SPA-resistant
Type-B	4307	2368	756	TA/SPA-resistant
Type-C	4713	2368	556	TA/SPA-resistant
Type-D	5879	2368	667	TA/SPA/DPA-resistant

3 Implementation Results

Table 2 shows the area and performance of the implementations of the four architectures. The performance of the types with micro-code ROM are better than Type-A. This is due to the reduction of the communication overhead between the MALU and MicroBlaze. A disadvantage is that the micro-code ROM costs extra area. In the case of Type-C, because the key can be stored in the coprocessor, the whole sequence of scalar multiplication can be performed in the coprocessor. Type-D is based on Type-C but with randomized key, which is longer than the real key. As a result, Type-D needs more clock cycles to finish one scalar multiplication and more area, namely 564 slices for the RNG [4].

4 Conclusions and Future Work

We proposed four types of architecture and their cost and performance. In the algorithm level, we implemented the Montgomery scalar multiplication to obtain TA/SPA resistance. Furthermore, by storing the key in the coprocessor, we only need to secure the coprocessor part. We also implemented a DPA-resistant architecture where the key is shielded by the RNG. As future work, we will mount circuit level countermeasures against DPA and measure the power consumption in order to find the best trade-off between area, performance and security.

References

[1] K. Sakiyama, B. Preneel, and I. Verbauwhede. A fast dual-field modular arithmetic logic unit and its hardware imlementation. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'06)*, pages 787-790, 2006.

[2] K. Tiri and I. Verbauwhede. A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *Proc. Design, Automation and Test in Europe Conference (DATE)*, pages 246-251, February 2004.

[3] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Ç.K. Koç and C. Paar, editors, *Proc. 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1717 in LNCS, pages 292-302, August 12-13 1999.

[4] D. Schellekens, B. Preneel, and I. Verbauwhede. Fpga vendor agnostic true random number generator. In *Proc. 16th International Conference on Field Programmable Logic and Applications*, pages 139–144, 2006.

Low-weight Joint Window Nonadjacent Form

Wei-Hua He and Fei-Ming Juan

Department of Computer and Information Science, Soochow University, Taiwan
whhe@cis.scu.edu.tw

Abstract

We propose a minimal joint window nonadjacent form that is useful for elliptic curve digital signature verification process. The average joint Hamming density among the joint window nonadjacent form representations with windows of width 4 is asymptotically 29.5%.

Keywords. digital signature, joint window nonadjacent form, joint Hamming density

1 Introduction

The form of $kP+lQ$ is frequently used in elliptic curve digital signature verification process, where P, Q are points on the curve, and k, l are scalars. Both scalars k and l are first recoded into the binary representation. Then, bits 0 and 1 correspond to a point doubling and addition operation, respectively. The main goal is to find a representation in which nonzero digits are as less as possible. The well-known representation is the nonadjacent form (NAF).

Solinas [So01] proposed a joint sparse form (JSF) for computing $kP+lQ$. To reduce the nonzero column, the goal is to create zero columns $\begin{smallmatrix}0\\0\end{smallmatrix}$ s in JSF as many as possible. About half of columns are zero. Its average joint Hamming density is 1/2 using the digit set $\{0, \pm 1\}$. It requires two pre-computed points $P + Q$, $P - Q$. The negative point for each point on the elliptic curve is obtained just by changing the sign of the y-coordinate. This fact can be exploited to reduce the number of pre-computed points. Recently, Dahmen *et al.* [DOT07] proposed a recoding algorithm that uses the digit set $\{0, \pm 1, \pm 3\}$ and requires ten pre-computed points. Its average joint Hamming density is 36.15%.

We will propose a new signed digit representation to reduce the joint Hamming density for simultaneous scalar multiplication. Our new signed digit representation is called joint window nonadjacent form. Using Markov chains, the average joint Hamming density is asymptotically 29.5%.

2 New Representation of Two Scalars

In this section, we describe a new right-to-left recoding algorithm with windows of width 4 to construct the joint window nonadjacent form for a simultaneous scalar multiplication using the digit set $\{0, \pm 1, \pm 3, \pm 4\}$. From Table 1, it can be seen that the decimal numbers between 5 and 11 are converted to $X00X$, where X represents a non-zero digit. The other decimal numbers in Table 1 can be converted to $000X$. Each window column is scanned from right to left and staring from the next non-zero digit. One of JWNAF representations in Table 1 will be applied.

In the recoding process, if one of both decimal numbers of four bits in each window column is between 5 and 11, then both most significant bits of that window column will be recoded again. This means that both most significant bits will be incorporated into the next window column. The proposed recoding process consists of 3 states $S_0 \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $S_1 \begin{pmatrix} 00X \\ 00X \end{pmatrix}$, and $S_2 \begin{pmatrix} 000X \\ 000X \end{pmatrix}$.

Each entry $P_{ij} = P(S_i \mapsto S_j)$ in the transition matrix $\begin{bmatrix} \frac{1}{4} & \frac{17}{32} & \frac{7}{32} \\ 0 & \frac{17}{24} & \frac{7}{24} \\ \frac{1}{4} & \frac{17}{32} & \frac{7}{32} \end{bmatrix}$ represents the transition probability, where $S_i \mapsto S_j$ represents from state S_i to state S_j. The steady state probability $(\pi_0,\pi_1,\pi_2)=(\frac{7}{79}, \frac{51}{79}, \frac{21}{79})$ is obtained from the transition matrix. Therefore, the average joint Hamming density is $\frac{0\times\frac{7}{79}+1\times\frac{51}{79}+1\times\frac{21}{79}}{1\times\frac{7}{79}+3\times\frac{51}{79}+4\times\frac{21}{79}} = 29.5\%$.

Table 1: JWNAF representations

Decimal	Binary	JWNAF	Decimal	Binary	JWNAF
1	0001	0001	5	0101	$100\bar{3}$
2	0010	0002	6	0110	$100\bar{2}$
3	0011	0003	7	0111	$100\bar{1}$
4	0100	0004	8	1000	1000
12	1100	$1000\bar{4}$	9	1001	1001
13	1101	$1000\bar{3}$	10	1010	1002
14	1110	$1000\bar{2}$	11	1011	1003
15	1111	$1000\bar{1}$			

The JWNAF requires 26 pre-computed points $3Q, 3P, P+Q, P+2Q, P+3Q, P+4Q, 2P+Q, 2P+3Q, 3P+Q, 3P+2Q, 3P+3Q, 3P+4Q, 4P+Q, 4P+3Q, -P+Q, -P+2Q, -P+3Q, -P+4Q, -2P+Q, -2P+3Q, -3P+Q, -3P+2Q, -3P+3Q, -3P+4Q, -4P+Q, -4P+3Q$. It does not need to store points $iP+jQ$ for all $i, j \in \{0, \pm1, \pm3, \pm4\}$, since both i and j should not be even.

3 Conclusions

The average joint Hamming density of the proposed JWNAF is asymptotically 29.5%. Our JWNAF will increase performance of simultaneous scalar multiplication on an elliptic curve. Future research in this topic includes a further reduction of the average joint Hamming density and the recoding time.

Acknowledgements

This work was supported in part by the National Science Council of Taiwan under grants NSC 95-2221-E-011-019, and by Taiwan Information Security Center (TWISC), National Science Council under the grants NSC 95-2218-E-001-001, and NSC95-2218-E-011-015.

References

[DOT07] Erik Dahmen. Katsuyuki Okeya and Tsuyoshi Takag. A new upper bound for the minimal density of joint representations in elliptic curve cryptosystems. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E90-A (5) 952-959, MAY 2007.

[So01] Jerome A. Solinas. Low-weight binary representations for pairs of integers. Technical Report CORR 2001-41, University of Waterloo, available at `http://www.cacr.math.uwaterloo.ca/techreports/2001/corr2001-41.ps`, 2001.

BDD-based Cryptanalysis of Hardware-oriented Stream Ciphers

Dirk Stegemann

Theoretical Computer Science, University of Mannheim, Germany
`http://th.informatik.uni-mannheim.de`
`dstegema@th.informatik.uni-mannheim.de`

Abstract

The main application of stream ciphers is online-encryption of arbitrarily long data. Many practically used and intensively discussed hardware oriented stream ciphers consist of a small number of feedback shift registers (FSRs) and a compression function that transforms the bit-streams produced by the FSRs into the output keystream. A generic cryptanalysis technique for this design is the Binary Decision Diagram (BDD) based recovery of the cipher's initial state. In contrast to other generic methods such as correlation attacks and algebraic attacks, the BDD-attack is able to reover the initial state from the shortest information theoretically possible amount of keystream. In this talk, we explain the basic structure of the attack, show how to reduce its memory requirements, and sketch some recent developments.

Keywords. Stream cipher, cryptanalysis, BDD

1 FSR-based Keystream Generators

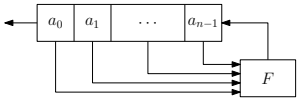
The main purpose of stream ciphers is online encryption of bitstreams $p \in \{0,1\}^*$ that have to be sent over an insecure channel. The output keystream $z \in \{0,1\}^*$ of the keystream generator is bitwise XORed to the plaintext stream p in order to obtain the ciphertext stream $c \in \{0,1\}^*$, i.e., $c_i = p_i \oplus z_i$ for all i. Based on a secret key K and a public initialization vector (IV), which have to be exchanged between sender and legal receiver in advance, the receiver can compute the keystream z in the same way as the sender computed it and decrypt the message using the above rule.

A keystream generator generally consists of an n-bit internal state $s = (s_0, \ldots, s_{n-1})$, a state update function $F : \{0,1\}^n \to \{0,1\}^n$ and an output function $C : \{0,1\}^n \to \{0,1\}^*$. The starting state s^0 of the generator is derived from K and the IV. At each clock t, output bits are produced according to $C(s^t)$ from the current state s^t, and the internal state is updated to $s^{t+1} = F(s^t)$. Hence, the output of the generator is completely determined by the starting state s^0.

A widely adopted architecture especially for hardware oriented keystream generators is to represent the internal state in a number of Feedback Shift Registers (FSRs) $R^1, \ldots, R^{k-1}$ of lengths $n^{(0)}, \ldots, n^{(k-1)}$. The state s of the generator is the combined state of the FSRs, hence $n = \sum_{i=0}^{k-1} n^{(i)}$. Among the many types of FSRs discussed in the literature, Linear and Non-linear Feedback Shift Registers have turned out to be particularly suitable building blocks for keystream generators.

Definition 1.1 *A* Feedback Shift Register (FSR) *consists of an n-bit register $a = (a_0, \ldots, a_{n-1})$ and a state update function $F : \{0,1\}^n \to \{0,1\}$. Starting from an initial configuration a^0, in each clock a_0 is produced as output and the register is updated according to $a := (a_1, \ldots, a_{n-2}, F(a_0, \ldots, a_{n-1}))$. Depending on whether F is a linear function, we call the register a* Linear Feedback Shift Register (LFSR) *or a* Nonlinear Feedback Shift Register (NFSR).

Figure 1: Feedback Shift Register (FSR) of length n



The FSR-construction is illustrated in Fig. 1. Similarly to a single FSR, we can think of an FSR-based keystream generator with k registers as producing an internal bitstream $(w_t)_{t\geq 0}$, where $w_t := w_{s(t)}^{r(t)}$ with $r(t) = t \bmod k$ and $s(t) = s \operatorname{div} k$, i.e., the t-th internal bit of the generator corresponds to the $s(t)$-th bit in the bitstream produced by $R^{r(t)}$. Examples for the FSR-based design include the E_0 cipher used in Bluetooth [1] and the newly proposed ciphers Trivium [2] and Grain [3].

Two important parameters of an FSR-based keystream generator are its best-case compression ratio γ, i.e., at most γm keystream bits can be produced from an internal state of length m, and the information ratio α, which denotes the average information that an observed piece of keystream reveals about the underlying internal bitstream.

2 BDD-based Cryptanalysis

The BDD-based attack on keystream generators, which was first introduced in [4], is a known-plaintext initial state recovery attack, i.e., the attacker tries to reconstruct the unknown initial state s^0 of the keystream generator from a few known plaintext bits and their encryptions. Since a ciphertext bit c_i is computed from a plaintext bit p_i and a keystream bit z_i via $c_i = p_i \oplus z_i$, the keystream bit z_i can be reconstructed from (p_i, c_i) by computing $p_i \oplus c_i = z_i$. We observe that for any internal bitstream $w \in \{0,1\}^m$ that yields a prefix of the observed keystream, w must be consistent with the update relations imposed by the FSRs, and the application of the output function to w must yield a prefix of the observed keystream. The main idea of the BDD-attack is to dynamically compute the internal bitstreams fulfilling these conditions and to represent the intermediate results with BDDs. Under certain pseudorandomness conditions, the attack is able to recover the initial state of size n in $2^{\frac{p(1-\alpha)}{p+\alpha}n}$ time and memory units from the first $\lceil \gamma\alpha^{-1}n \rceil$ consecutive bits of keystream, where p denotes a small cipher-dependent parameter.

References

[1] The Bluetooth SIG. *Specification of the Bluetooth System*, February 2001.

[2] Christophe de Canniere and Bart Preneel. Trivium specifications. eSTREAM, ECRYPT Stream Cipher Project, 2005. `http://www.ecrypt.eu.org/stream`.

[3] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A stream cipher proposal: Grain-128. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/010, 2005. `http://www.ecrypt.eu.org/stream`.

[4] Matthias Krause. BDD-based cryptanalysis of keystream generators. In *Proc. of EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 222–237. Springer, 2002.

[5] Matthias Krause and Dirk Stegemann. Reducing the space complexity of BDD-based attacks on keystream generators. In *Proc. of FSE 2006*, volume 4047 of *LNCS*, pages 163–178. Springer, 2006.

Preimages for Reduced-Round Tiger[1]

Sebastiaan Indesteege[2] and Bart Preneel

Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
{`sebastiaan.indesteege,bart.preneel`}`@esat.kuleuven.be`

Abstract

The cryptanalysis of the cryptographic hash function Tiger [AB96] has, until now, focussed on finding collisions [KL06, M+06]. In this paper we describe a preimage attack on the compression function of Tiger-12 (*i.e.* Tiger reduced to 12 rounds out of 24) with a complexity of $2^{63.5}$ compression function evaluations.

We show how this can be used to construct second preimages (complexity $2^{63.5}$) and first preimages (complexity $2^{64.5}$) for Tiger-12. These attacks can also be extended to Tiger-13 at the expense of an additional factor 2^{64} in complexity.

Keywords. Tiger, hash functions, preimages

1 Introduction

Tiger [AB96] is an iterative cryptographic hash function, designed by Anderson and Biham in 1996, with an output size of 192 bits. The state size is 192 bits and each iteration uses a message block of 512 bits. Tiger's compression function consists of 24 rounds of a state update transformation, each using a 64-bit message word X_i to update three 64-bit chaining variables. After 8 rounds, a key schedule algorithm is used to compute the next 8 message words from the previous ones. Finally, the initial values are fed forward. For a complete description of Tiger, we refer to [AB96].

Attacks on Tiger have so far focussed on finding collisions. Kelsey and Lucks [KL06] found a collision attack on 16 rounds reduced Tiger with a complexity of 2^{44} compression function evaluations. Mendel *et al.* [M+06] extended this to a collision attack on 19 rounds of Tiger with a complexity of 2^{62} compression function evaluations. In both papers some weaker attacks for a larger number of rounds (*e.g.* pseudo-collisions) were also shown.

In section 2 we will show a preimage attack on the compression function of Tiger-12. Based on this, we will construct first and second preimages for Tiger-12 in section 3. Finally, we conclude.

2 Preimages for the compression function of Tiger-12

In this section, we describe a method to find preimages for the compression function of Tiger, reduced to 12 rounds. We will use the notations from [KL06, M+06].

1. We make arbitrary choices for X_8, X_9, X_{10} and X_{11}, and guess Y_7. Using the key schedule, we can now determine the values of Y_0, Y_1, Y_2, Y_3, X_1, X_2 and X_3 such that, if the guess for Y_7 is correct, it is guaranteed that the chosen values for X_8 through X_{11} will appear from the key schedule.

[1] This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

[2] F.W.O. Research Assistant, Fund for Scientific Research – Flanders (Belgium).

2. We choose X_7 (there are 2^{64} choices) and compute X_0 using the key schedule.
3. Using the state update transformation – in both directions – we can now determine A_3, B_3, C_3, A_6, B_6 and C_6. Continuing in the backwards direction, even A_5, B_5 and B_4 can be determined at this point.
4. Now, $X_4 = C_3 \oplus B_4$. Knowing X_4, we can compute A_4 and C_4. Then $X_5 = C_4 \oplus B_5$, which enables us to compute C_5. Finally, $X_6 = C_5 \oplus B_6$.
5. Using the key schedule, we verify if the guess for Y_7 is correct. If it is, we have found a preimage. If not, we make a different choice for X_7 and try again.

The probability that the guess for Y_7 is correct is 2^{-64} so we expect to find a preimage after 2^{64} tries. Note that one try requires only about 2/3 of the computations of one evaluation of the compression function. Hence, the overall complexity of the attack is equivalent to about $2^{63.5}$ evaluations of the compression function.

The attack can be extended to 13 rounds, by guessing the value of X_{12} before the attack and verifying if the guess was correct afterwards. This again happens with a probability of 2^{-64}, yielding a total complexity of $2^{127.5}$. An extension to 14 rounds has a complexity that is only marginally below that of a brute force attack.

3 First and second preimages for Tiger-12

Using the attack to find second preimages for Tiger-12 is straightforward, for (padded) messages with at least two message blocks. The preimage will have the same length as the given message. All but the last two message blocks can be chosen arbitrarily. The last message block, containing the padding, is copied. Then the attack is applied to the second to last message block.

Finding first preimages is a bit more involved due to the padding. First we choose the message length such that only a single bit of padding will be placed in X_6 of the last message block. X_7 of this message block contains the message length as a 64-bit integer. As X_7 is now fixed a priori, we will use the freedom in the choice of Y_7 instead during the attack. Because of this, a larger part of the key schedule has to be redone on every try. Also, we have to verify if the last bit of X_6 is a 1, as dictated by the padding rule. This happens with probability 2^{-1}, resulting in an overall complexity of $2^{64.5}$.

Of course an extension to Tiger-13 can be made as explained in section 2. The complexities become $2^{127.5}$ for second preimages and $2^{128.5}$ for first preimages.

4 Conclusion

In this paper we have shown a preimage attack on the compression function of Tiger-12 and Tiger-13 (complexity $2^{63.5}$ resp. $2^{127.5}$). This can be used to construct first and second preimages for the Tiger-12 and Tiger-13 hash functions.

References

[AB96] Ross Anderson and Eli Biham. Tiger: A Fast New Hash Function. In *Fast Software Encryption, Proceedings*, LNCS vol. 1039, pp. 89–97, Springer, 1996.

[KL06] John Kelsey and Stefan Lucks. Collisions and Near-Collisions for Reduced-Round Tiger. In *Fast Software Encryption, Proceedings*, LNCS vol. 4047, pp. 111–125, Springer, 2006.

[M+06] Florian Mendel, Bart Preneel, Vincent Rijmen, Hirotaka Yoshida and Dai Watanabe. Update on Tiger. In *Progress in Cryptology - INDOCRYPT 2006, Proceedings*, LNCS vol. 4329, pp. 63–79, Springer, 2006.

Algebraic Attacks on Block Ciphers with Several Plaintext-Ciphertext Pairs

Nicolas T.Courtois[1] and Blandine Debraize[2,3]

[1] University College of London, Gower Street, London, UK,
[2] Gemalto, Meudon, France and [3] University of Versailles, France
blandine.debraize@laposte.net

Abstract

In this paper we show that algebraic attacks on block ciphers are in many cases more efficient if several plaintext-ciphertext pairs are used. We make computations on a toycipher to show it, an we study the link between the efficiency of this type of attack and the structure of the implicit equations describing the S-boxes being part of the design of block ciphers.

Keywords. block ciphers, algebraic attack, S-box, quadratic equations

1 Introduction

Algebraic attacks on block ciphers [1, 4] study different ways of describing the problem of recovering the secret key as a system of multivariate equations. When Groebner bases algorithms like F4 are used to solve these systems, the complexity of these attacks strongly depends on the maximal degree of the polynomials occurring during the computations. It is well known that it is possible to decrease the degree of a system of equations by adding new variables. Using several plaintext-ciphertext pairs in algebraic attacks increase the number of variables in the systems of equations but decrease the time complexity in some cases. In this paper we build a toy block cipher in order to make computations to prove this assertion and to study the mechanisms of the efficiency of algebraic attacks using several plaintext-ciphertext pairs.

2 Description of our Toycipher

Our toycipher is very simple. It is called ToyBlock. Several versions are possible. The non linear layer of each round is made of B 4 bits parallel S-boxes. Each S-box is the S-box number 2 of the block cipher Serpent, see [2]. The key schedule is a simple permutation of the key bits: $K_{i,j} = K_{0,(j+i \mod (4.B))}$, where K_0 is the secret key. Each of the r rounds consists in:

- A bitwise XOR of the current state Z_i with the derived key K_i.
- A layer of B parallel 4 bits S-boxes.
- A simple linear layer that is not described here.

We call the version with r rounds and B parallel S-boxes ToyBlock(B,r). We write the system of GF(2) equations describing the problem of the key recovery of this toycipher by writing the implicit quadratic equations of the S-box and the linear equations of the linear layers.

3 Using Several Plaintext-Ciphertext Pairs

We used our implementation of the algorithm F4 and the F4 version of the computer algebra system Magma to perform computations on ToyBlock, with degree reverse lexicographical monomial ordering.

The F4 computations for the polynomial systems describing one plaintext-ciphertext pair of respectively ToyBlock(4,3) and ToyBlock(6,3) could not be performed at degree 3. This means

that a degree at least 4 is necessary to solve the systems. This leads to a theoretical complexity of respectively 2^{53} and 2^{73} for these ciphers. Indeed, we were not able to recover the key for both systems by applying F4 on them. However we were able to solve at degree $d = 2$ the systems describing 32 plaintext-ciphertext pairs of ToyBlock(3,4) in 3 minutes and to solve the system describing 64 plaintext-ciphertext pairs of ToyBlock(3,6) in 51.54 minutes. Of course exhaustive search is much faster in both cases but we have proven that using several plaintext-ciphertext pairs can considerably decrease d and the time complexity of this type of attack.

4 Implications of S-box Equations in these Attacks

We show that some special equations that we call X^2 equations have implications in the fact that using several plaintext-ciphertext pairs is useful to perform algebraic attacks on ToyBlock. An X^2 equations is an implicit quadratic equations describing the S-box, such that its quadratic monomials only mix input variables.

Two of the implicit equations describing the ToyBlock S-box are X^2 equations. We compare this S-box to a 4 bits reduction of the Rijndael S-box proposed in [3]. None of the equations describing this second S-box is an X^2 equation. We show that new linear equations appear during the computation of F4 at degree 2 on several plaintext-ciphertext pairs of ToyBlock, whereas no linear equation appear when the ToyBlock S-box is replaced by the reduction of the Rijndael S-box of [3].

5 Chosen Plaintext Guess-and-determine Attacks

We show that using several plaintext-ciphertext pairs especially improves the complexity of guess-and-determine algebraic attacks. In this type of attacks, several key bits are guessed and their values substituted in the system of equations before the system is solved by a Groebner bases algorithm. By carefully choosing the plaintexts of the pairs, the complexity is still decreased. With this method we were able to break ToyBlock (32,5) only at degree 2.

Conclusion

In this paper we show that using several plaintext-ciphertext pairs improves the complexity of algebraic attacks on certain block ciphers. This improvements are due to special properties of the GF(2) equations describing the S-boxes. The Serpent S-boxes have these properties whereas the Rijndael S-box does not have. They especially allow to decrease the complexities of guess-and-determine algebraic attacks.

The degree 2 properties of the S-boxes that we have studied in this paper can be extended at higher degree. Their study would be an interesting further research.

References

[1] Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Asiacrypt 2002, LNCS 2501, pp.267-287, Springer.

[2] Ross Anderson, Eli Biham, Lars Knudsen: *Serpent, a flexible Block Cipher With Maximum Assurance* First AES Candidate Conference, 1998, Ventura California, available at www.cl.cam.ac.uk/ rja14/serpent.html

[3] Carlos Cid, Sean Murphy, Matthew J. B. Robshaw: *Small Scale Variants of the AES*.FSE 2005: 145-162

[4] Johannes Buchmann, Andrei Pyshkin, Ralf-Philipp Weinmann: *Block Ciphers Sensitive to Groebner Basis Attacks*.CT-RSA 2006: 313-331

Lattice-based partial key exposure attacks on signature schemes

Paul Kubwalo*, John Mulagha*, and Edward F. Schaefer†

* Department of Mathematics, Mzuzu University, Private Bag 201, Luwinga, 2 Mzuzu, Malawi
johnmulagha@yahoo.com

† Department of Mathematics and Computer Science, Santa Clara University, Santa Clara, CA 95053, USA
eschaefer@scu.edu

Abstract

For three digital signature schemes (DSA, Schnorr, ElGamal), there is a modular equation involving a private key, a secret session key, and some known quantities. We adapt Coppersmith's lattice attack to find solutions of this equation. We show that if the total number of consecutive unknown bits in the two keys is less than the number of bits in the modulus, then there is a fast algorithm that, in practice, finds these two secret keys. In the former two schemes, this modulus is the smaller of the two primes, currently around 160 bits.

Keywords. discrete logarithm problem, digital signature

1 Extended abstract

Lattice-based techniques have been used to quickly solve for an RSA decrypting key d when a certain fraction of the bits of d are known; see [BD, BDF, Co, HG]. Such ideas were originally due to Coppersmith. In this article, we present such an attack for three signature systems whose security is based on the difficulty of the finite field discrete logarithm problem.

In some signature/authentication cryptosystems (the Digital Signature Algorithm, Schnorr signature scheme, Schnorr authentication scheme, ElGamal signature scheme) there is a linear equation modulo a prime q. For the DSA and Schnorr algorithms, q is the smaller of the two primes given; currently q is chosen to have around 160 bits. This modular equation has two unknowns. These unknowns can represent the unknown consecutive bits in the secret key and the random session key. We will show that if the product of the two unknowns is less than q, then similar lattice based techniques and the LLL algorithm enable us to quickly determine those unknowns and compromise the cryptosystem. In other words, assume that q has b bits. We can consider the private key and the session key to be b-bit numbers as well. Let us assume that the total number of consecutive bits in the secret key and session key, known to the attacker, is greater than b. Then under normal circumstances, the attacker can solve for the secret key and session key in polynomial time. Some of the bits could be known because of some unrelated attack, like a timing attack, a brute force attack or exploiting a faulty pseudo-random number generator.

In this paper, we first describe the signature and authentication schemes and create the modular equations we will solve. We then prove a theorem showing that if an equation of the form $y + cx + d = 0 (\mathrm{mod}\, q)$ has a solution (x_0, y_0) with $|x_0 y_0| < q$ then under normal circumstances we can find two polynomials in two variables, with integer coefficients, that are 0 at (x_0, y_0). In practice, we can use resultants to determine these unknowns. An immediate consequence of the theorem is that if the secret key and session key are both so small (perhaps

in order to speed up modular exponentiations) that their product is less q, then we should be able to quickly solve for them.

We use the theorem to prove a similar result when the total number of unknown consecutive bits in the secret and session keys is less than the number of bits in q. We then give the results of computer experiments. For example, if $q \approx 2^{160}$ and the total number of unknown consecutive bits is less than 128, then we can solve the discrete logarithm problem in under 1 minute.

To conclude we give a lower bound for how small the secret and private keys can be. We also recommend a future direction of research, namely combining this result with that in [HS].

References

[BD] Dan Boheh and Glenn Durfee. Cryptanalysis of RSA with Private Key d Less Than $N^{0.292}$. *IEEE Transactions on Information Theory,* IT-46(4):1339–1349, July 2000.

[BDF] Dan Boneh, Glenn Durfee and Yair Frankel. An attack on RSA given a small fraction of the private key bits. In proceedings AsiaCrypt '98, Lecture Notes in Computer Science, Springer-Verlag, 1514:25–34, 25–34, 1998.

[Co] Dan Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10:233-260, 1997.

[HG] Nick Howgrave-Graham. Finding small roots of univariate modular equations revisited. In porceedings *Cryptography and Coding*, Lecture Notes in Computer Science, Springer-Verlag, 1355:131–142, 1997.

[HS] Nick Howgrave-Graham and Nigel Smart. Lattice Attacks on Digital Signature Schemes. *Designs, Codes and Cryptography*, 23(3):283–290, 2001.

Template Attacks On Masking: An Interpretation

Benedikt Gierlichs[1] and Kåre Janussen[2]

K.U. Leuven, ESAT/SCD-COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{bgierlic,kjanusse}@esat.kuleuven.be

Abstract

In [OM07] four flavors of a Template Attack against a software implementation of the AES, which is protected by the masking countermeasure, are presented. The results lead to a ranking of the four attack styles with respect to experimental success efficiency, measured in the number of traces required to break the implementation. However, no attempt is made to explain the cause of the observations. In this work we analyze the first of the afore mentioned attacks from an information theoretic point of view in order to fill this gap. Our analysis explains the experimental observations, allows to derive a lower bound of the number of traces required, and also points out a reasoning. Simulations indicate the correctness of our result.

Keywords. Template Attack, Boolean Masking, Information Theory.

1 Introduction

Side channel attacks are the most promising approach to reveal secret data from embedded cryptographic devices. Due to visions like Pervasive Computing or Ambient Intelligence such devices, *e.g.* smartcards, mobile phones, PDAs, RFIDs, and sensor nodes, become more and more present in everyday life which in turn stresses the need for secure devices. In the last decade, much research has been conducted on side channel attacks and countermeasures. The first side channel to be discovered was the timing behavior [PK96] of devices, later followed by power dissipation [KJJ99], and electromagnetic radiation [QS01]. However, much research focuses on the power side channel as it is probably the most practical one to exploit, if accessible.

The power consumption side channel exists due to a deterministic link between power consumption and processed data, caused by standard CMOS technology. All countermeasures against power analysis aim at either weakening, hiding, or destroying this link. For example, the masking countermeasure [AG01, CG00] at algorithm level aims at destroying the link between data values and processed values, which in turn de-correlates physical observables from data values. On the other hand, more powerful attacks were developed to break countermeasures such as masking [TM00].

Recently, the efficiency of four flavors of Template Attacks against masked implementations has been assessed in [OM07]. The comparison leads to an efficiency ranking of the attacks with respect to the number of power traces required to recover the key, but no attempt is made to explain these experimental results. In this paper we aim at filling this gap by analyzing the first of the four attacks from an information theoretic point of view. Our approach does not only allow to explain the experimental results but also to estimate a lower bound of the number of traces required.

The rest of the paper is organized as follows. Section 2 introduces basic notions of information theory and our notation. Section 3 summarizes the attack's principles while Section 4 provides our analysis and results. We conclude our work in Section 5.

[1] This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by FWO projects G.0475.05 and G.0450.04, by the European Commission FP6 MIRG project SESOC, number MIRG-CT-2004-516568, and by the K.U. Leuven-BOF.

[2] This work was supported in part by Rejselegat for Matematikere, Denmark and K.U. Leuven.

2 Preliminaries

We recapitulate[3] basic notions of information theory and show how to apply them to fundamental distributions. Further we introduce our notation and bring forward a simple reduction.

2.1 Information theoretic measures for selected distributions

Let X be a random variable on a the discrete space $\mathcal{X} = \{0, 1, \ldots, 255\}$ with probability distribution $\mathbb{P}_X = \left\{\frac{1}{256}, \ldots, \frac{1}{256}\right\}$, hence uniformly distributed. The entropy of X is

$$\mathsf{H}(X) = -\sum_{x \in \mathcal{X}} \mathbb{P}_X[X = x] \log_2 \mathbb{P}_X[X = x] = 8 \text{ [bits]}. \tag{1}$$

The conditional entropy $\mathsf{H}(X|Y)$ and the mutual information $\mathrm{I}(X;Y)$ are defined as follows:

$$\begin{aligned} \mathsf{H}(X|Y) &= -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \mathbb{P}_{X,Y}[X = x, Y = y] \log_2 \mathbb{P}_{X|Y}[X = x|Y = y] \\ \mathrm{I}(X;Y) &= \mathsf{H}(X) - \mathsf{H}(X|Y) = \mathsf{H}(X) + \mathsf{H}(Y) - \mathsf{H}(X,Y) = \mathrm{I}(Y;X). \end{aligned} \tag{2}$$

The Hamming Weight function, denoted by $\mathrm{HW}(\cdot)$, maps a bit string x to the number of bits in the string $x = x_1 x_2 \cdots x_8$ which are set to '1'. The resulting Hamming weight of a byte value is $\mathrm{HW}(x) = \sum_{i=1}^{8} x_i$. Let X be a uniform random variable of 8-bit strings, then $\mathrm{HW}(X)$ is a random variable denoting the Hamming weight of X. It follows that $\mathrm{HW}(X)$ takes values in $\mathcal{W} = \{0, 1, \ldots, 8\}$ with the Gaussian probability distribution

$$\mathbb{P}_{\mathrm{HW}(X)} = \left\{\tfrac{1}{256}, \tfrac{8}{256}, \tfrac{28}{256}, \tfrac{56}{256}, \tfrac{70}{256}, \tfrac{56}{256}, \tfrac{28}{256}, \tfrac{8}{256}, \tfrac{1}{256}\right\} \tag{3}$$

which can be found by evaluating the binomial coefficients $\binom{8}{w}$ for $w \in \mathcal{W}$. The entropy of $\mathrm{HW}(X)$ according to the probability distribution in (3) is

$$\mathsf{H}(\mathrm{HW}(X)) = -\sum_{w \in \mathcal{W}} \mathbb{P}_{\mathrm{HW}(X)}[\mathrm{HW}(X) = w] \log_2 \mathbb{P}_{\mathrm{HW}(X)}[\mathrm{HW}(X) = w] = 2.5442 \text{ [bits]}. \tag{4}$$

2.2 Notation and reduction

The plaintext, the encryption key, and the mask are 8-bit values x, k, and m of the random variable X, K, and M respectively. K and M are uniformly distributed over the space $\{0, 1, \ldots, 255\}$ with probabilities $\mathbb{P}_K(k) = \mathbb{P}_M(m) = \frac{1}{256}$. While k is fixed, m is only constant in each encryption round after which it is regenerated. Both are unknown to the attacker who is trying to determine k, while x is known. The AES Sbox function is referred to as $\mathrm{Sbox} : \{0,1\}^8 \to \{0,1\}^8$ as a permutation on 8-bit strings. The intermediate results v_m are 8-bit values from the set $\{0, 1, \ldots, 255\}$, defined as $v_m = \mathrm{Sbox}(x \oplus k) \oplus m$. The intermediate result v_m can be observed as a realization of the random variable V_m, which is uniform over $\{0, 1, \ldots, 255\}$.

In the remainder of this paper we are going to make use of a reduction in the way we treat the computations of entropy of the random variable V_m. In this known-plaintext attack, x is assumed to be known and we want to show $\mathsf{H}(\mathrm{HW}(V_m) \mid X) = \mathsf{H}(\mathrm{HW}(K))$.

When x is known, $\mathrm{Sbox}(x \oplus k)$ can be viewed as a random permutation on input k, outputting a random byte which makes the entity $\mathrm{Sbox}(x \oplus k) \oplus m$ also a random byte upon which we conclude that $\mathsf{H}(\mathrm{HW}(V_m) \mid X) = \mathsf{H}\left(\mathrm{HW}\left(\mathrm{Sbox}(X \oplus K) \oplus M\right) \middle| X\right) = \mathsf{H}(\mathrm{HW}(K))$. In the remainder the known-plaintext property is implied. Hence, for the sake of clarity we will write the above equation as $\mathsf{H}(\mathrm{HW}(V_m)) = \mathsf{H}(\mathrm{HW}(K))$.

[3] See [CT05] for a more thorough treatment of the subject.

3 The attack

In this attack (cf. Sect. 3.1 in [OM07]), the data words m and $v_m = \mathrm{Sbox}(x \oplus k) \oplus m$ are targeted. The experiments reported in [OM07] were carried out on a 8051 compatible microcontroller, which as the authors say, leaks the Hamming weight of the 8-bit data words processed. Taking this fact as granted, we can model the side channel leakage of the microcontroller as depicted in Fig. 1. The figure also illustrates our notation. It seems plausible and feasible to build templates

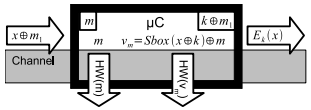


Figure 1: Schematic illustration of the side channel

for each leaked Hamming weight of intermediate values such that, during classification, the Hamming weight of those data words can be reliably and non-ambiguously detected from a single power trace. Henceforth, this will be assumed. So during profiling in this case, the adversary builds 81 templates covering all possible combinations of the 9 Hamming weights for m and the 9 Hamming weights for v_m.

In the attack phase, an adversary extracts the two Hamming weights $\mathrm{HW}(m)$ and $\mathrm{HW}(v_m)$ from a given measurement t_i by template classification, which in turn allows him to assign key candidates k' with posteriori probabilities $p(k' \mid t_i)$ (cf. (2, 4) in [OM07]), illustrated in Fig. 2. This process is iterated with fresh measurements and the posteriori probabilities of the key

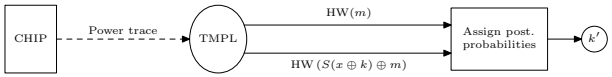


Figure 2: Template Attack on masking

candidates are updated each time (cf. (3) in [OM07]). The number of iterations depends on the number of measurements that are available and on the key extraction accuracy the adversary desires. Frankly, these two restrictions are antipodal in the sense that higher accuracy requires more measurements. From their experiments, the authors of [OM07] concluded that about 15 measurements are sufficient for reliable key recovery.

4 Theoretical analysis

When evaluating the power of this attack, the central question we want to answer is:

> *What is the remaining uncertainty on the key given a pair of Hamming weights* $\mathrm{HW}(m)$ *and* $\mathrm{HW}(v_m)$*?*

Putting it in information theoretical terms using the previously defined notation, what is $\mathsf{H}(K) - \mathsf{H}(K \mid \mathrm{HW}(M), \mathrm{HW}(V_m))$? The answer is by the definition (2) exactly $\mathrm{I}(K; \mathrm{HW}(M), \mathrm{HW}(v_m))$.

In the following we will show how to compute the mutual information after the first observation. Then we will indicate how to update the remaining entropies in order to compute the mutual information of subsequent observations.

To calculate the mutual information I, we first need to derive the three entropies $\mathsf{H}(K)$, $\mathsf{H}(\mathrm{HW}(M), \mathrm{HW}(V_m))$, and $\mathsf{H}(K, \mathrm{HW}(M), \mathrm{HW}(V_m))$. By definition of K and using Eq. (1), its

entropy is $\mathsf{H}(K) = 8$ [bits]. Since the random variables M and $V_m = \text{Sbox}(X \oplus K) \oplus M$ are statistically independent, also $\text{HW}(M)$ and $\text{HW}(V_m)$ are independent. Hence, using (4) yields[4],

$$\mathsf{H}(\text{HW}(M), \text{HW}(V_m)) = \mathsf{H}(\text{HW}(M)) + \mathsf{H}(\text{HW}(V_m)) = 2.5442 + 2.5442 = 5.0884 \text{ [bits]}. \quad (5)$$

Finally, we need to compute $\mathsf{H}(K, \text{HW}(M), \text{HW}(V_m))$, which in practice may be done by counting the different joint occurrences over all possible scenarios (k = 0...255, m = 0...255). We have computed the entropy $\mathsf{H}(K, \text{HW}(M), \text{HW}(V_m)) = 11.938$ [bits]. Hence, for the first observation,

$$\text{I}(K; \text{HW}(M), \text{HW}(V_m)) = 8 + 5.0884 - 11.938 = 1.1499 \text{ [bits]},$$

which means that in the average case, an adversary will gain 1.1499 bits of information on the key[5]. Before computing the mutual information for the second observation, we have to update (reduce) the key entropy according to the information gained. We now have $\mathsf{H}(K) = 8 - 1.15 = 6.85$ [bits].

At this point the analysis faces a numerical problem, because binomial coefficients of the form $\binom{n}{k}$ are only defined for $n \geq k \in \mathbb{N} \cup \{0\}$. In other words, if the key entropy is 6.85 bits then the question is whether the remaining $2^{6.85} = 115.37$ key candidates are interpreted to be 115 or 116. We decide to analyze both scenarios which represent the worst and the best case from the adversaries point of view and hence will always use $\lceil 2^{\mathsf{H}(\cdot)} \rceil$ (worst case) and $\lfloor 2^{\mathsf{H}(\cdot)} \rfloor$ (best case) in our computations. Doing so allows us to provide an estimation of a lower bound.

Applying the above mentioned, we assume the number of remaining key candidates to be 116 in the worst case (115 in the best case) after the first observation and compute the entropies $\mathsf{H}(\text{HW}(M), \text{HW}(V_m)) = 5.0656$ [bits] (worst) and 5.0646 (best), and $\mathsf{H}(K, \text{HW}(M), \text{HW}(V_m)) = 10.799$ [bits] (worst) and 10.786 (best). Conclusively, the mutual information after the second observation is $\text{I}(K; \text{HW}(M), \text{HW}(V_m)) = 1.1166$ [bits] (worst) and 1.1286 (best).

Table 1: Information and entropy values for 8 observations

Obs.	$\mathsf{H}(K)$		$\mathsf{H}(\text{HW}(M), \text{HW}(V_m))$		$\mathsf{H}(K, \text{HW}(M), \text{HW}(V_m))$		$\text{I}(K; \text{HW}(M), \text{HW}(V_m))$	
	worst	best	worst	best	worst	best	worst	best
1	8	8	5.0884	5.0884	11.938	11.938	1.1499	1.1499
2	6.8501	6.8501	5.0656	5.0646	10.799	10.786	1.1166	1.1286
3	5.7335	5.7215	5.0083	5.0047	9.6363	9.5760	1.1055	1.1502
4	4.6280	4.5713	4.9115	4.8933	8.4159	8.2803	1.1237	1.1843
5	3.5043	3.3870	4.7788	4.7210	7.2113	6.8893	1.0718	1.2187
6	2.4326	2.1683	4.5543	4.4765	6.0101	5.3177	0.97674	1.3271
7	1.4558	0.8413	4.0641	2.5442	4.7308	2.5442	0.78917	0.8413
8	0.6667	0	3.9954	2.5442	3.9954	2.5442	0.66667	0
9	0	0	2.5442	2.5442	2.5442	2.5442	0	0

Table 1 shows the three occurring entropy values and the mutual information for 8 observations, after which the key entropy is entirely removed and the correct key revealed, as indicated in the ninth line (eighth for best case). This result deviates from the result presented in [OM07]. They state that about 15 traces are needed. The deviation and hence the sub-optimal performance of their key recovery can be directly explained from the developing of their key probability plots, see Fig. 3 in [OM07]. The probability for the correct key candidate decreases at least twice which happened likely due to incorrect template classification. Our model does not cover such imprecisions, which are very much dependent on a specific scenario (implementation, measurement setup, etc.) and very difficult to model in a general approach. Further, our model does not cover redundant information as for example the (unlikely) event that a pair of plaintext and mask occur twice.

[4]Using (3), this can also be computed from a probability matrix $\mathbb{P}_{\text{HW}(M),\text{HW}(V_m)} = \mathbb{P}_{\text{HW}(M)} \times \mathbb{P}_{\text{HW}(V_m)}$ with entrances $\mathbb{P}_{i,j} = \mathbb{P}_{\text{HW}(M)}(\text{HW}(m) = i - 1) \cdot \mathbb{P}_{\text{HW}(V_m)}(\text{HW}(v_m) = j - 1)$.

[5]Not literately learning 1.1499 actual key bits, but rather 1.1499 bits of non-trivial Shannon information [CS48].

To verify the correctness of our result, we have simulated the attack 10.000 times under the same ideal conditions, *i.e.* correct Hamming weight detection and no redundancy. For each simulation a fresh key byte was chosen at random from a uniform distribution. The simulation showed an average attack duration of 8.5 rounds ($\sigma = 1.87$, median = 8 rounds). Table 2 shows how the number of remaining key candidates evolves in the simulation and in the worst/best case analytic scenarios.

Table 2: Remaining key candidates for best/worst case and simulation

best case	256	115	52	23	10	4	1	1	1
worst case	256	116	54	25	12	6	3	2	1
simulation	256	118	56	27	13	7	4	2	1

Ideally, one would expect the simulation result to be bound by the best and worst case analytic results, which is not the case for our experiment. However, the small difference between simulation and analytic result can be explained by the number of our simulations. In order to observe an unbiased average behavior, we would need to assure that we simulate all combinations of plaintext and mask for each possible key byte equidistributed. Finally, it is important to notice that the average attack duration has been estimated correctly.

5 Conclusion

We have analyzed a Template Attack on a masked software implementation of the AES from an information theoretic point of view. Our analysis provides a lower bound on the number of traces required in the average case[6]. Our simulations indicate the correctness of the results. As future work we will map the three remaining flavors of Template Attacks on Masking described in [OM07] to our framework and analyze their efficiency.

References

[OM07] E. Oswald, S. Mangard, Template Attacks on Masking, In Proceedings of CT-RSA 07, pp. 243-256, LNCS 4377, Springer Verlag

[CT05] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley and Sons, 2005

[CS48] C. Shannon, A Mathematical Theory of Communication, Bell System Technical Journal, 1948

[PK96] P. Kocher, Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems, In Advances in Cryptology - Crypto 96, pp. 104 - 113, Springer Verlag

[KJJ99] P. Kocher, J. Jaffe, B. Jun, Differential Power Analysis, In Advances in Cryptography - Crypto 99, pp. 388-397, LNCS 1666, Springer Verlag

[QS01] J.-J. Quisquater, D. Samyde, ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards, In Proceedings Smart Card Programming and Security, LNCS 2140, Springer Verlag, 2001

[AG01] M. Akkar, C. Giraud, An implementation of DES and AES secure against some attacks, In proceedings of CHES 01, LNCS 2162, Springer Verlag

[CG00] J. Coron, L. Goubin, On boolean and arithmetic masking against differential power analysis, In proceedings of CHES 00, LNCS 1965, Springer Verlag

[TM00] T. S. Messerges, Using Second-Order Power Analysis to Attack DPA Resistant Software, In proceedings of CHES 00, LNCS 1965, Springer Verlag

[6]The expected number of traces required is often used as an indicator of the level of security.

Wibree Security — towards better energy efficiency

Jan-Erik Ekberg
Nokia Research Center

1 Introduction

Personal-area-networking (PAN) and peer-to-peer communication over unlicenced radio communication channels is already widely used in static installations like keyboard to PC or mobile phone to wireless headset. The Bluetooth [10] standard is presently the dominant solution for PAN communication but other technologies like ZigBee address specific market segments, like home automation.

Resource consumption and especially battery drain is the foremost communication concern in PANs. This drain can more or less be mapped to baseband activity — sleep periods should be optimized, and channel initiation and control overhead in relation to actual transmitted data kept as small as possible. As wide-spread solutions today do not live up to the promise of "battery efficiency", new solutions like Wibree are needed to bridge the gap between PAN and sensor networks.

In personal and sensor networks the transmitted data is to an overwhelming degree private and often confidential, and a secure mode on the link itself is motivated since many services are designed to run directly on top of the radio. The security context and needs of an OSI layer 2 radio technology is to most parts a given. Early standardization efforts[1] already identify the needed security services for a link layer to be confidentiality and integrity as well as data origin authentication. Key management and context setup needs complete the picture.

The security solutions chosen for Wibree are mainly driven by efficiency. The primary goal has been not to waste transmitted bits on the channel, and a security overhead (measured in bits) of close to 0% for a session is better than any other standardized solution yet. Another important design criteria has been cost - this can mainly be mapped to a desire to keep algorithm complexity as well as the security overhead in terms of added logic in the stack to a reasonable minimum.

2 Energy efficiency

Energy consumption is a cost issue that is especially prominent in battery-operated devices, and indirectly very noticable to the end user. For security protocols, this cost category can be split into *computational cost*, *memory cost* and *communication cost*. We will see that all of these need to be minimized to achieve an efficient end result.

As for communication cost, [8] and [2] give the energy needed to transmit a single bit as $18\mu J/bit$ and $50\mu J/bit$, in their respective solution. As transmission range, speed, power control and modulation as well as channel overhead like retransmissions and synchronization to a high degree will affect the bits/J ratio for a radio, these values can at best be considered to be real-world indications of the range of transmission power consumption. In the comparisons below, the $18\mu J/bit$ measure is used. Sending and receiving seems to be roughly on par energywise - [6], [13] and [2] all indicate that transmission is only around twice as costly as receiving, or even less.

[1] The IEEE 802.10 security standard is a good example of a consortium that appropriately identified the security goals of the link layer

Regarding memory energy consumption, the following deductions can be made. Both [1] and [7] cite the relative energy consumption of the flip-flop memory included in their respective cryptographic chips running at 100kHz and 3.3V / 1.5V, and the result gives a memory power consumption of 22.8nW/bit and 9nW/bit respectively. Additionally, microprocessor cache energy consumption has been measured on an experimental basis in [5] — an appropriate example for memory with high content volatility. Many technologies and speeds were compared, but one measure at gigabit clock speeds and 0.75V put the energy consumption at 19.5 nW/bit, which is comparable to the flip-flop measures above. Thus, for a 1kB memory the last measurement amounts to 1.6 $mW = mJ/s$, i.e. energy-wise equivalent to transmitting 9 bits/s in the scenario outlined here.

The power consumption of a few common cryptographic algorithms optimized for energy were collected from references into table 1. The SW measurements [12], were measured on a Compaq iPaq H3670, with an Intel SA-1110 StrongARM clocked at 206MHz, i.e. on a PDA device. As energy consumption seems to increase more the 100-fold when going from HW to SW instantiations, algorithms implemented in SW should clearly be avoided for frequently occuring calulations.

Algorithm	Energy/op (HW)	Energy/op (SW)
AES(128b)	0.045 μJ [7]	17.9 μJ
RSA(1024b)	2.41/0.37 mJ [9]	546/16 mJ
ECC(163b)	0.66/1.1 mJ [4]	134/196 mJ

Table 1: Power consumption for some crypto algorithms)

Table 1 collects the data in this section for comparison. The insight from this data serves as a basis for the design chosen for Wibree.

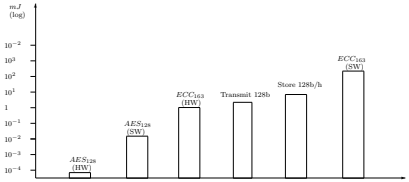


Figure 1: Energy consumption compared (logarithmic scale)

3 Security for the Wibree link layer

The radio link signalling in Wibree is strictly stateful. A peer-to-peer flush operation does not exist, and the link breaks if state is lost. This possible shortcoming is counterbalanced by methods to achieve fast reconnects. The absence of complex radio state management also makes it easy to integrate security mechanisms to the link in an efficient manner.

For confidentiality, AES-128 in counter mode has been chosen. The selection is based on the general acceptance of the algorithm, easy HW block availability as well as decent size- and

performance parameters in hardware. Using counter mode makes it possible to leave out the decryption part of the cipher block for optimization, and the mode carries no data overhead caused by block padding. Known shortcomings of the counter mode include that the mode is sensitive to key/counter - reuse, and that it does not provide integrity protection even with a authentication code embedded in the plaintext[2].

Integrity is thus provided by combining a checksum to the counter mode in a manner contextually very similar to AES-GCM [11]. The key material for the keyed checksum is taken from the first AES block in a message (which is reserved for that purpose), but instead of using a key checksum based on multiplication in a small Galois field, Wibree uses a integrity check value (ICV) based on polynomial division in a similar (GF(2)) field [3]. The chosen method is an extension of a typical CRC calculation – essentially extended with a varying divisor polynomial[3] and a final transformation. The hardware CRC block is trivially extended in this manner (see fig. 2), and thus the same block can calculate both CRCs (fixed polynomial and identity transformation) and keyed ICVs. For encrypted connections we now replace the CRC with the ICV, and to achieve the needed real-time performance the ICV keys for a packet protected with ICV are calculated and applied in advance for the next expected packet. Combining the radio and cryptographic checksum in this way achieves transmission savings for every packet, but more importantly the error state handling can be combined. An acknowledged packet on the link implies that the transmission has succeeded also with respect to the integrity check.

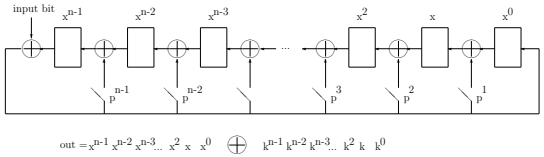


Figure 2: ICV (checksum) calculation replacing CRC

The 128-bit counter for the AES CTR mode includes an IV which is fixed for one session, and computed as a side-result of the session key calculation, a direction-bit, a block subcounter used within a single packet, and a 48-bit packet counter. As the link is non-windowed and the reception of every packet is acknowledged there is no space for packet loss and the counter need not be included in the PDU. Thus neither the cipher, counter nor the integrity check induce any communication overhead compared to plaintext transmission.

Session key setup is optimized for speed, and included as an integral part of the setup phase for a secure link[4]. The 48-bit address field of the connecting device is used as the random key material contribution of that device[5], and the target device will contribute 80 random bits with a specific host-layer PDU that can be pre-inserted into the link-layer for transmission at the earliest possible

[2]Bit-flipping attacks in a known-plaintext context are possible

[3]the polynomial is selected from the result of an AES block with a minimum of precaution, i.e. the x^0 term should be set to 1 to protect against pathological polynomials

[4]With the exception of pairing sessions, sessions are either encrypted or not from the start of the session. Switching between secure and non-secure mode using a 3-way handshake is only supported as a link-layer service for the protocol stack, not as a feature

[5]the identification of the connecting party in a secure link is based on a specific diversifier, see section on keying

occasion. The resulting 128-bit of random material will be used both to diversify a session key and to initialize a fixed portion of the encryption counter. It has been estimated that the security setup (the session key generation) will add only around 10ms to the setup time of a session.

4 Keying

Wibree keys fall into two sets - keys used for encryption, and keys used for privacy – the Wibree standard supports a mechanism to protect against address tracking while still providing reachability. All keys in Wibree are 128-bit randomly generated or diversified AES keys, as the host stack size is mimimized by extensive re-use of the AES HW block in all management, key set-up and privacy host operations. To save size and computation cycles, no other security algorithms have been deployed in the specification.

The keying in Wibree is taking into account the presumed imbalance in capability between host and slave devices - e.g. sensors will often have limited or no persistent storage. In such a model *distributory* long-term keying has advantages compared to *contributory* systems - the device that distributes the keys can do so based on diversification of a local master secret. This operation can be repeated for every session, eliminating the need for persistent storage for a pairwise secret.

5 Performance measures

The efficiency of the solution can be measured in several domains. The only cryptographic elements used is a HW random number generator, and AES (+ ICV) implemented in hardware. As these to necessary parts are exposed from the link layer to the host layer, the energy efficiency of the solution in the algorithmic domain should be close to optimal given the chosen algorithm.

A non-optimized test implementation of the security-related logic in the host layer (key management, privacy functions) in ANSI C (i386) compiles to 4000 bytes (of ROM) and needs 200 bytes (RAM) for heap+stack, again compact at least with respect to R/W memory needs.

The communication overhead for security compared to non-encrypted communication is at best (if no transmission errors occur) 4 link layer packets with a payload of 11 bytes in total. Additionally 2 bytes in the connection request PDU can be attributed to security alone. As session traffic carries no encryption overhead, these 4 packets and 13 bytes per session is the complete security transmission overhead. For comparison, table 2 collects the percentual overhead of transmitted security parameters in a typical data payload packet for different radio communication technologies. The overhead is calculated for a payload size of 16 bytes (sensor-like traffic).

6 Conclusions and future work

The Wibree radio technology is targeted towards data communication between small, possibly mobile, battery-operated devices with cost, size and battery constraints. The security architecture defined for the radio technology bridges these constraints with functionality needed to guarantee confidentiality and integrity for the data transmitted, as well as privacy protction for the users of the devices. Efficiency has been achieved with careful design and algorithm selection, an by tightly integrating bulk encryption algorithms and security session control with the radio control plane.

Protocol	Sync.	IVs etc.	MAC	Overhead
GPRS/2G	yes	0	24	3B / 19%
WCDMA/3G	no	11	48	7+B /45%
Bluetooth	yes	0	16	2B/12%
802.11 WLAN	no	64	96	20B/125%
802.15.4 ZigBee	no	40	48	11B/69%
TinySec (sensors)	no	16	32	6B/38%
Wibree	yes	0	0	0B/0%

Table 2: Percentual overhead of security algorithms - examples

References

[1] M Feldhofer & al. A case against currently used hash functions in rfid protocols. Workshop on RFID Security 2006 - RFIDSec06, July 13-14, Graz, Austria.

[2] Miller & al. Minimizing energy consumption in sensor networks using a wakeup radio. In *Wireless Communications and Networking Conference*, 2004.

[3] Scott A. Vanstone Alfred J. Menezes, Paul C. van Oorschot. *Handbook of Applied Cryptography*. CRC Press, 1996.

[4] Guido Bertoni, Luca Breveglieri, and Matteo Venturi. Ecc hardware coprocessors for 8-bit systems and power consumption considerations. *itng*, 0:573–574, 2006.

[5] H Hanson; S Hrishikesh; V Agarwal; S Keckler; D Burger. Static energy reduction techniques for microprocessor caches. *IEEE Transactions on VLSI Systems*, 11(3), June 2003.

[6] M. Feeney, L.M.; Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE , vol.3, no.pp.1548-1557 vol.3*, 2001.

[7] V. Feldhofer M.; Wolkerstorfer J.; Rijmen. Aes implementation on a grain of sand. In *Information Security, IEE Proceedings , vol.152, no.1pp. 13- 20, Oct.*, 2005.

[8] A Haowen Chan; Perrig. Pike: Peer intermediaries for key establishment in sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE , vol.1, no.pp. 524- 535 vol. 1, 13-17 March*, 2005.

[9] A.P. Charndrakasan J Goodman. An energy-efficient reconfigurable public-key cryptography processor. *IEEE Journal of Solid-state Circuits*, 36(11), Nov 2001.

[10] Charles F Sturman Jennifer Bray. *Bluetooth: Connect Without Cables*. Prentice Hall, 2000.

[11] D A McGrew and John Viega. The galois/counter mode of operation (gcm), may 31, 2005.

[12] Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha. Analyzing the energy consumption of security protocols. In *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design*, pages 30–35, New York, NY, USA, 2003. ACM Press.

[13] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180, New York, NY, USA, 2003. ACM Press.

A new Security Model and a Generic Construction for CCA2-Secure Certificateless Cryptosystems

Ewan Fleischmann

University of Mannheim
http://th.informatik.uni-mannheim.de
ewan.fleischmann@gmx.de

Abstract

The concept of certificateless public key cryptography was presented at AsiaCrypt 2003 by Al-Riyami. It has been proposed as an attractive alternative to certificate-based and identity-based encryption schemes. It combines the implicit public key authentication of an identity-based scheme with the escrow-free property of a certificate-based one. However, security proofs in the standard model seem still to be very difficult. The (classic) CCA2 security model requires a challenger to decrypt a ciphertext without knowing the private key. A solution – as e.g. proposed in [DK05] – might be to relax this requirement. This should not result into a feasible attack. Another problem is the high complexity of the applied model – its description suffers from a massive list of capabilities by simultaneously listing prohibited parameter configurations. This leads to a very unhandy work environment.

In Section 2 we present a security model which is fully equivalent to the well known CCA2 model but has a shorter description. Furthermore, if we relax the challenger decryption requirement, the advantages of the new description even gets more evident.

In Section 3 we give a generic method for creating certificateless schemes by combining an identity-based and a public-key scheme. If the latter two are CCA2-secure, then the new certificateless scheme is also CCA2-secure (in the above mentioned weaker sense).

Keywords. certificateless public key cryptography, security model, attack model, generic construction

1 Introduction

1.1 Certificateless PKC Basics

In certificate-based public key cryptosystems an entity's public-key is generated from some random information unrelated to his identity. So it has to be certified by a third party (e.g. with a certificate issued by a certification authority). Any participant who wants to use a public-key must first verify the corresponding certificate to check the validity of the public-key. Certificate-based public key cryptosystems require a large amount of storage and computing time to verify and revoke certificates.

The notion of identity-based cryptography (ID-PKC) was introduced by Shamir in 1984: the public-key of a user can be derived from his unique identifier (e.g. email-address). But the first practical implementation was due to Boneh and Franklin [BF03] in 2003 . ID-PKC eliminates the certificates and greatly simplifies the key management. However, an inherent problem of ID-PKC is key escrow, i.e. the private-key of each user is known to a third party who can decrypt any ciphertext and forge signatures on any message for any user. Moreover, ID-PKC requires a secure channel between users and the third party to deliver the private keys since the users can not generate them (generation of private keys requires an additional master-key only known by the third party). Due to all of these problems, it seems that ID-PKC should

be considered being suitable only for (small) private or corporate networks with lower security requirements.

To alleviate these problems Al-Riyami and Paterson [AL03] introduced the concept of certificateless public key cryptography (CL-PKC). The private key is not any more generated by a third party – called key generation center (KGC) – alone (unlike in ID-PKC schemes) but does also require a *partial* private key which is an additional user-chosen secret. In this way, they successfully eliminate the built-in escrow of ID-PKC schemes by still not requiring a certificate.

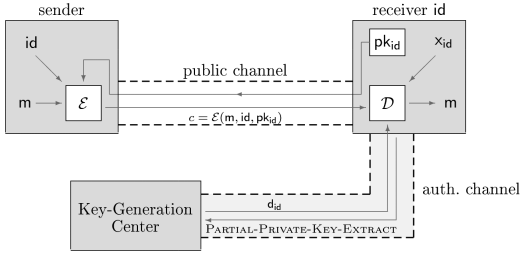


Figure 1: Communication model - certificateless public key cryptography

According to [AL03, DK05] we define a certificateless public-key encryption scheme by seven probabilistic algorithms: Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Encrypt and Decrypt.

1.2 Security of CL-PKC schemes

The security of certificateless' schemes is assessed in terms of two different attackers:

(I) The first kind of attacker (called type I) is meant to represent a normal third party attack against the system. Here, an entity in possession of all the public keys attempts to break the IND-CCA2 security of the scheme. Due to the uncertified nature of the public-keys produced by the users, we must assume that an attacker is able to replace these entities' public keys at will.

(II) The second kind of attacker (called type II) represents a malicious key generation center, who may not replace entities' public keys.

The original definitions of type I and type II attacker were based upon an oracle model containing five different queries. Additionally a type I attacker has four different oracle query constraints, a type II attacker has five different constraints (see e.g. definitions in [AL03, DK05]). Some of them are quite hard to understand.

2 An Equivalent and Realistic Security Model

2.1 An Equivalent Model

Now we describe an fully equivalent, but easier to handle, security model for CL-PKC schemes. It only utilizes four oracles and has less query constraints. The core idea is simply to remove the **R**eplace**P**ublic**K**ey oracle call and substitute it with an additional parameter (a public key) in some of the remaining four oracles.

Our new model is based upon the following oracles:

1. **P**rivate**K**ey**E**xtract($\mathsf{id} \in \{0,1\}^*$) returns $\{\mathsf{s_{id}} \in \mathcal{K}_{priv}\}$
   Call SET-PRIVATE-KEY and return the received value to the caller.
2. **P**artial**P**rivate**K**ey**E**xtract($\mathsf{id} \in \{0,1\}^*$) returns $\{\mathsf{d_{id}} \in \mathcal{K}_{partial}\}$
   Call PARTIAL-PRIVATE-KEY-EXTRACT and return the received value to the caller.
3. **R**equest**P**ublic**K**ey($\mathsf{id} \in \{0,1\}^*$) returns $\{\mathsf{pk_{id}} \in \mathcal{K}_{pub}\}$
   If called the first time generate a public key using SET-PUBLIC-KEY. If needed call SET-SECRET-VALUE before.
4. **D**ecrypt($\mathsf{id} \in \{0,1\}^*, \mathsf{pk_{id}} \in \mathcal{K}_{pub}, \mathsf{c} \in \mathcal{C}$) returns $\{\mathsf{m} \in \mathcal{M}\}$
   If $\mathsf{pk_{id}}$ = **R**equest**P**ublic**K**ey($\mathsf{pk_{id}}$) then generate a private key with SET-PRIVATE-KEY. Decrypt the ciphertext c by calling m =DECRYPT and return m to the caller.

($\mathcal{K}_{priv}$ denotes the set of private keys, $\mathcal{K}_{partial}$ the set of partial private keys (only known by the individual user), $\mathcal{K}_{pub}$ the set of public keys, $\mathcal{M}$ is the set of messages.)

DEFINITION 2.1 ***(Attackers).***
Any probablistic, polynomial-time type I (type II) attacker $\mathcal{A}^{\mathcal{I}}$ ($\mathcal{A}^{\mathcal{II}}$) should have negligible advantage in winning the following IND-CCA2 game.

Init: *Challenger calls* SETUP.
(type I): The system parameters are passed on to $\mathcal{A}^{\mathcal{I}}$, the masterkey is kept private.
(type II): The system parameters and the masterkey are passed to $\mathcal{A}^{\mathcal{II}}$.

Phase 1: *Attacker calls oracles (adaptive).*

Challenge: *Attacker returns two equal length plaintexts, an (challenge) identity and a (challenge) public key (only type I) to the challenger. For the (challenge) identity, the attacker must not have asked for the private or the partial private key during phase 1. The challenger choses randomly one of the two plaintexts and encrypts it using the given identity and the public key. The resulting challenge ciphertext is returned to the attacker.*

Phase 2: *Attacker calls oracles (adaptive).*

Guess: *Attacker outputs a guess which of the two plaintexts was encrypted.*

The oracle queries in phase 1 and 2 are subject to the following constraints:

1. ***Type I Attacker:***
   (a) *The attacker must not ask for the private key of the challenge identity.*
   (b) *The attacker must not ask for the decryption of the challenge ciphertext using the challenge identity and challenge public-key.*
   (c) *If **P**artial**P**rivate**K**ey**E**xtract($\mathsf{id} \in \{0,1\}^*$) is called, then the following condition must be true:*
   *id $\neq$ challenge id $\vee$ challenge public key = **R**equest**P**ublic**K**ey(challenge id).*
2. ***Type II Attacker:***
   (a) *see Type I (a)*
   (b) *see Type I (b)*
   (c) *For every oracle query the condition:*
   *public key = **R**equest**P**ublic**K**ey(id)*
   must hold (i.e. public key is always bound to the identity).

With this new definition we have three main advantages:

1. Easier to understand. This is due to the explicit nature of the public key replacement.
2. One oracle less (four instad of five).

3. Less number of constraints. Especially the non-trivial restricitions were reduced from 2 in the original security model to one here (Type I, (c)).

PROOF. (Sketch)
The equivalence of the two security models is shown as follows: If a CL-PKC scheme is secure in our new model $\Leftrightarrow$ it is secure in the (classic) model. The approach is, as usual, indirect.
(skipped a lot of detail...) □

2.2 An Equivalent and Realistic Model

In [DK05] it is shown that in the security model (as presented e.g. in [AL03]) the security against a type I attacker implies the existence of a type II attacker. Currently there are only two known workarounds: proofs in the random oracle model or weakening the security model. The security model can be weakened by eliminating the fore mentioned condition that the challenger must be able to decrypt a message without knowledge of the private key. We change the Decrypt oracle query as follows:

Decrypt($\mathsf{id} \in \{0,1\}^*, \mathsf{x_{id}}, \mathsf{c} \in \mathcal{C}$) returns $\{\mathsf{m} \in \mathcal{M}\}$
The $\mathsf{x_{id}}$ represents the user chosen secret thus the challenger can calculate the private key using SET-PRIVATE-KEY. If $x_{id} = \perp$ the decryption is performed for the normal public key (as e.g returned by the **R**equest**P**ublic**K**ey oracle). Decrypt the ciphertext c by calling m =DECRYPT and return m to the caller.

Remark: This is – in our new security model – the only change needed to adapt to the new weaker model. In the old model – as e.g. demonstrated in [DK05] – one more restriction has to be added for type I attackers. So the new model has only three restrictions, the old five (respectively for type I and type II attackers).

3 Generic Construction

Cryptographic schemes seldom get developed from scratch. Often it is more convenient and reasonable to create them combining well-known ones. We will create a certificateless scheme by a combination of a public key and an identity-based scheme. In [YL04] Yum and Lee presented a generic construction based upon a cascade. It was broken in 2006 by D. Galindo [GA06]. Our scheme is created as a *tuple* construction. The plaintext is divided into two (equal length) components. One of them is encrypted with an identity-based scheme (IBE), the other with a public key (PKE) scheme. In general, this will lead to a ciphertext which is about doubled in length. The plaintext can only be recreated if the two (decrypted) components are combined again. Thus we need a function SPLIT which divides our plaintext and COMBINE which recombines the two parts again. ENCRYPT and DECRYPT are defined as follows:

ENCRYPT(m)
(1) (m_{pke}, m_{ibe}) = SPLIT(m)
(2) c_{pke} = PKE.ENCRYPT(m_{pke})
(3) c_{ibe} = IBE.ENCRYPT(m_{ibe})
(4) return $c = (c_{pke}, c_{ibe})$

DECRYPT(c)
(1) m_{pke} = PKE.DECRYPT(c_{pke})
(2) m_{ibe} = IBE.DECRYPT(c_{ibe})
(3) m = COMBINE(m_{pke}, m_{ibe})
(4) return m

The function SPLIT and COMBINE are dependent on the type of plaintext. If the message space is e.g. equal to $\{0,1\}^n$ then SPLIT choses a random value $r \in \{0,1\}^n$ and calculates $m_{ibe} = r$, $m_{pke} = r \oplus m$. SPLIT calculates the message as $m = m_{ibe} \oplus m_{pke} = m$.

If the message space is a group (or field) then one can easily chose a random element r and let the operation $\oplus$ be a group (or field) operation. Combine uses the same operation with r^{-1}.

One severe problem is, that the IBE and PKE scheme both have to use the same message space (or SPLIT and COMBINE care for this inconvenience). But there do exist construction methods for creating an IBE- and PKE scheme which have this feature (e.g. [CHK04]).

Theorem 3.1 *The CL-PKE scheme resulting from the tuple construction is IND-CCA2 secure (in the weaker sense) if PKE is a IND-CCA2 secure public-key scheme and IBE is a IND-CCA2 identity-based scheme.*

PROOF. (Sketch)

1. CL-PKE can be broken by type I attacker ⇒ PKE or IBE are unsecure in contradiction to the assumption.
2. CL-PKE can be broken by type II attacker ⇒ PKE or IBE are unsecure in contradiction to the assumption.

(skipped again a lot of detail...) □

References

[AL03] Al-Riyami and Paterson, Certificateless Public Key Cryptography, ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology, LNCS, Springer-Verlag, 2003

[BF03] Boneh and Franklin, Identity-Based Encryption from the Weil Pairing, SICOMP: SIAM Journal on Computing vol. 32, 2003

[CHK04] Canetti and Halevi and Katz, Chosen-Ciphertext Security from Identity-Based Encryption, EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT, 2004

[DK05] A. Dent and C. Kudla, On Proofs of Security for Certificateless Cryptosystems, http://citeseer.ist.psu.edu/dent05proofs.html, 2005

[GA06] David Galindo and Paz Morillo and Carla Ràfols, Breaking Yum and Lee Generic Constructions of Certificate-Less and Certificate-Based Encryption Schemes, 2006, EuroPKI (Springer LNCS) pp. 81-91, http://dx.doi.org/10.1007/11774716_7

[YL04] Yum and Lee, Generic Construction of Certificateless Signature, ACISP: Information Security and Privacy: Australasian Conference, 2004

Encryption Based Trust Negotiation

Dalia Khader

University of Bath
ddk20@bath.ac.uk

Abstract

In this work we define two types of trust negotiation, Encryption Based or Signature Based, depending on how the credentials are created. We also suggest a new way of building an Encryption Based Scheme. We propose a Trust Model that could allow us to mix between Encryption Based Schemes depending on the properties needed in the negotiation and that model provides the negotiation with strong properties such as flexibility, privacy,...etc. We also introduce possible protocol problems when using Encryption Based Signature.

Keywords. Automated Trust Negotiation, Hidden Credentials, IBE, Traitor Tracing, Broadcast Encryption

1 Introduction

Trust negotiation is the process of exchanging digital credentials between strangers in order to built trust gradually. Digital credentials are the computer analog to paper credentials such as Student ID, Driving license, passports,...etc. In old systems authentication simply relied on previous knowledge about the other party such as having to use passwords. Nowadays, in open systems where two strangers need to communicate, property based authentication and/or authorization is needed and using trust negotiation is ideal[WSJ00]. Trust Negotiation protocols describe the exact method of exchanging the credentials.
Trust negotiation schemes studied until this day could be divided into two categories according to what cryptography algorithms where used in creating the credentials and policies. The first is Signature-Based Credentials (SBC) where digital credentials are obtained from a credential issuer that signs them and anyone could verify them later. Credential issuers are usually institutions and/or organization (Ex. University, Government, Company...etc). The second category is an Encryption-Based Credentials (EBC) as in Hidden Credential Scheme (HC-Scheme)[HBSO03]. HC-scheme is based on identity based encryption. The credential issuer in such a case has templates for credentials that could be used as a public key. One party encrypts the resource with its policies (policies are generated from the public keys) and the other authenticates itself to credential issuer in order to obtain the private key to decrypt. Another possible EBC is Hidden Credential with Traitor Tracing(HCTT) which we propose in this report. In this report we define Broadcast Encryption and show how we could use it in building HCTT schemes. We show a trust model for EBC schemes and protocols used along with such a model. We finally show some attacks on those protocols.

2 Using Broadcast Encryption and Traitor Tracing Algorithms

A Broadcast Encryption Scheme is a set of algorithms that allow a transmitter to send encrypted messages to a collection of users such that only a privileged subset of users can decrypt them[H03]. Broadcast Encryption was desperately needed for copyright purposes, digital TV

channels distribution, and music transmissions. Such an encryption scheme was first introduced in 1994 in[FN94]. In that paper the authors pictured a scenario where there is a center and a set of users that are provided by that center with keys. Now the center broadcast an encrypted message and that only subscribed users could decrypt. Nonmembers of such system are curious to get to know the message. If they succeed the system is said to be broken.
Broadcast Encryption included three main algorithms SETUP, BROADCAST, DECRYPT. SETUP outputs the keys of encryption and decryption. BROADCAST encrypts a message and sends to everyone. DECRYPT is an algorithm that enables valid user to decrypt the message being sent. Two extra algorithms where later introduced to the scheme TRACING and REVOKING. TRACING was first introduced by Chor, Fiat and Naor in[CFN94] . The idea of tracing is to be able to tell which user has betrayed the system. Betraying the system could be by either distributing the key to illegal decoders or by distributing the message after decoding it[NNL01]. REVOKING is an algorithm that removes legal users from the system and disables them from encrypting. Hidden Credentials with Traitor Tracing (HCTT) consists of four algorithms that follow directly from the public key traitor tracing algorithms as in[BF99]. There is a Credential Create algorithm, an Encryption algorithm, a Decryption algorithm and a traitor tracing algorithm.

- Create-Credential: is equivalent to the key generation algorithm. It generates a public key to every credential A_{pub} and for every person that satisfies the credential it generates a private key A_{owner}.
- Encrypt: Encrypts a resource that is guarded with a certain policy, where each policy is created with A_{pub} of the required credentials. It creates a ciphertext CT.
- Decrypt: Decrypts a CT with possible combination of A_{owner}. If A_{owner} is the corresponding private key to A_{pub}, which was used in creating CT then you get to the resource otherwise rubbish is returned.
- Tracing: Incase of suspicion of impersonating another person. Tracing is a deterministic algorithm that will give away the traitor.

The reason behind using broadcast in trust negotiation is to get rid of the overhead of exchanging IDs(as in[HBSO03]). Besides some Traitor Tracing algorithms enables revocation, and others are more efficient than the scheme in[HBSO03] in terms of the size of ciphertexts and keys. The disadvantage of HCTT is the need of tracing algorithm which is not required in[HBSO03].

3 Trust Model for EBC Negotiation Schemes

In this section we present a general trust model for EBC trust negotiation. We will start by defining the model. We will explain the protocols used in it and show the benefits of such a model.

3.1 An Overview of the Trust Model

The trust model we are using consists of four main components:

- Trust Agents(TA): A trust agent is a software on a persons device that does the trust negotiation on his behalf it consists of the database with users policy, a wallet that consists of users credentials, a library that contains different classes and programs that deal with different trust strategies, and last but not least a security agent that decides which strategy to use with which policy and when (See Figure 1).
- Trust Negotiation Protocols(TNP): is the protocol that two agents of different people use when negotiating and exchanging credentials

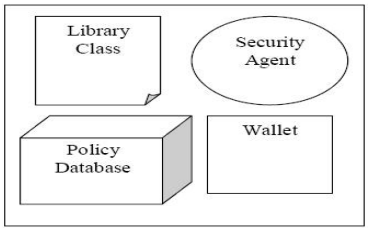


Figure 1: Trust Agent

- Credential Issuers (CI): are third parties that are contacted when a person wants to obtain a new credential or is setting new policies. It is considered an off-line third party since connections with it are not done as part of the trust negotiation but before.
- CI-Level Protocols(CILP): are protocols defined when obtaining new credentials or creating new policies.

There are three main interactions that occur in this model: Creating Policies, Obtaining Credentials, and Negotiating. To explain further we will use a real life scenario. The health center (HC) gives insurance (I) to oversea students. Home office in this case is a credential issuer and it decides to use EBC scheme based on a particular Encryption Scheme that has E and D to be its encryption and decryption algorithms relatively. Giving out the public key A_{pub} to present an oversea student visa and A_{priv_J} to every international student J.

- Creating policies: Every Trust Agent that is updating there policies' database need to contact the credential issuer to obtain the keys corresponding to each credential needed in the policy. With those keys the credential issuer will also send the trust agent a method for locking anything in their wallet with those policies. Method is saved in library class and policy is saved in the policy database with a link to the particular method and a link to an element in the wallet. Security qualities of the CILP protocols used between TA and CI is dependent on whether policy is secretive or not. Getting back to our scenario above, HC wants to create policy: If J=oversea-student, give(I). HC downloads A_{pub} and E from home office. HC saves E in library class. It creates I and saves it in wallet. It saves A_{pub}, a link to E and a link to I in policy database.
- Obtaining Credentials: Every trust agent that is trying to obtain a credential has to contact the CI. The CI will send him the credential as a key that will be able to unlock other trust agents policies that need this particular credential. The unlocking method is also downloaded from the CI to the trust agents library. That key is locked with certain policies from the policy database and kept in the wallet along with a link to the method in the library. Safe CILP protocols that guarantee no information leakage are needed in this case. So according to this the home office will send in a secure manner a private key A_{priv_J} to the student J along with D. The student saves D in the library class and A_{priv_J} with a link to D in the wallet.
- Negotiating: When two trust agents decide to begin negotiation they start exchanging locked credentials one at a time depending on what information they receive from the other trust agent. The security agent chooses when and with what credential to start. It also chooses when to terminate. Last but not least it chooses the strategy it wants to take through out the negotiation. It uses the different methods from the class library. The protocols between two agents should be done to ensure fair trade of information with no leakage of privacy to either parties. In our scenario, HC and student J are negotiating. The security agent links the policy database, wallet and library together. In HC the security agent calculates $E(A_{pub}, I)$ and sends it to student. The security agent in student could calculate $D(E(A_{pub}, I), A_{priv_J})$ to get I. Negotiating in such a matter preserves the privacy of the home office and the student. No one will get to know I without having a private key. At the same time health center would not be able to tell whether or not the other party is a student unless if they reply back with an obvious clue.

3.2 Advantages of the Trust model

In this section we will present some of the advantages of having such a trust model. We will explain how it solves the problems faced in SBC schemes.

- Supporting Complex Policies: A credential expression is a logical expression that serves to denote the combinations of credentials that satisfy a complex policy.
  Assume we have public keys $A_{C_1}, A_{C_2}, ..., A_{C_N}$ for credentials $C_1, C_2, ..., C_N$. Let R be the resource protected with a policy P. Let E and D be the encryption and decryption schemes. The following are examples of different complex policies.
  - $P : C_1$ is required;Send: $E(R, A_{C_1})$
  - $P : (C_1$ or $C_2)$;Send:$(E(R, A_{C_1}), E(R, A_{C_2}))$

  - $P : (C_1$ and $C_2)$;Send: $E(E(R, A_{C_1}), A_{C_2})$
  - $P : MofN(C_1, C_2, ..., C_N)$; Send:$MofN(E(R, A_{C_1}), E(R, A_{C_2}), ..., E(R, A_{C_N}))$
- Flexibility: The main reason for EBC to be more flexible than SBC schemes is that credentials are sent encrypted. Sophisticated protocols and strategies where introduced in SBC schemes to ensure security and privacy for the negotiators. The research at that time was about finding a way that an eavesdropper or even a malicious negotiator would not be able to get unnecessary information and at the same time guarantee that the negotiation is complete. A negotiation is complete if it does not terminate unless it is necessarily. Those complicated protocols and strategies were not compatible with existing data exchange protocols. They were also not compatible with one another, thus you can not easily change the negotiation strategy being used at that moment. All of these problems do not exist in EBC schemes because not having the decryption keys makes it very hard to reveal any information.
- Cyclic Policies: In some SBC schemes cyclic policies were solved when insuring a fair swap of certain credentials when non of the negotiators has to go first and give their information before the other with a risk of betrayal. Nevertheless, cyclic policies are not even a problem in EBC because the parties would not be actually exchanging those certain credentials but using them in encrypting and decrypting the resource.
- Preserving Privacy: In EBC schemes privacy is reserved in so many ways. The resource and the policy protecting it could not be revealed without owning the credentials that breaks the encrypted data apart. Whether or not the decryption took place is also kept private, in other words you would not be able to know if the other party managed to satisfy your policies. When you get a reply from the other party in which you could not break because you do not satisfy it, you would not be able to tell if that reply is for real and you dont have the required credentials or is it a rubbish to cover the fact that the person you are negotiating with was unsuccessful in breaking your policy. In conclusion privacy is preserved for both parties there is no way to guess policies, credentials or even resource unless you satisfy the policies protecting them.

3.3 Trust Models Protocols

There are two main protocols in our trust model (Section 3.1); one is between the negotiators (TLP), and the other is between a negotiator and a credential issuers (CILP). The latter is more about giving either public or private keys of a particular credential after authenticating the negotiator. Old authentication and access control methods take place before sending the keys needed in a secure manner. In this section we will focus more about the TLP protocols and demonstrate two possible attacks on them.
Assume Alice and Bob are two strangers[HBSO03]. Alice wants to request a resource from Bob, where both the request and resource are protected with policies. P_{REQ} is a policy in Alices policy database that is linked with the encryption algorithm E_1 in her library class and the request REQ in her wallet. P_{RES} is a policy in Bobs policy database that is linked with the encryption algorithm E_2 in his library class and the resource RES in his wallet. P_{SC} is a policy in Bobs policy database linked to the encryption algorithm E_3 in his library and some sensitive credential in his wallet that will be needed in decrypting Alices request. $E_4(M, s)$ is any symmetric encryption algorithm with s being the secret key and M being the plaintext.

1. Alice sends the request encrypted with P_{REQ} as shown:
   Alice $\overset{E_1(REQ, P_{REQ})}{\longrightarrow}$ Bob
2. Bob sends back the response encrypted in a symmetric encryption together with the decryption key being itself decrypted as shown:

Alice $\overset{E_3(E_2(s,P_{RES}),P_{SC}),E_4(Response,s)}{\longleftarrow}$ Bob [1]

Such a protocol has two different threats:

- Side Channel Attack:[HBSO03] In such a protocol Alice will be able to know whether Bob possess a credential or not from his behavior. If he fails to response back that is a clue that he does not satisfy her policy. This affects Bobs privacy. To prevent that from happening Bob should simply reply with rubbish in every time he is incapable of decrypting. Alice would not be able to distinguish between rubbish and encrypted data that she does not satisfy its policy.
- Cooperative Attack : Assume that in the previous scenario $P_{REQ} : C_1$ and C_2. Bob obtains the private key of C_1 and Alex has the private key for C_2. Let A_{C_1} and A_{C_2} be the public keys for C_1 and C_2 relatively. Neither Bob nor Alex are allowed access to Alices Resource but both could cooperate with each other and break into Alices Policy as follows:
  1. Alice sends the request encrypted with P_{REQ} as shown:
     Alice $\overset{E_1(E_1(REQ,A_{C_1}),A_{C_2})}{\longrightarrow}$ Bob
  2. Bob sends what he received to Alex:
     Bob $\overset{E_1(E_1(REQ,A_{C_1}),A_{C_2})}{\longrightarrow}$ Alex
  3. Alex decrypts what Bob sent him and sends the following:
     Alex $\overset{E_1(REQ,A_{C_1})}{\longrightarrow}$ Bob
  4. Bob could decrypt and get request REQ and continue the negotiation without Alice discovering anything.

  Someone might say that this kind of attack is hard to happen if we have the ID of the person implicit in the private keys he owns, once they are created by the CI(as in[HBSO03]). We would response that this is true if the ID is chosen in away that you could prove ownership of it.

4 Summary

Trust negotiation could be either EBC or SBC. HCTT is a possibly new EBC scheme. The trust model provides flexbility and privacy. It also supports complex policies and overcomes cyclic policies. Protocols of such a model have two main problems: side channel attack and cooperative attacks.

References

[WSJ00] W Winsborough and K Seamons and V Jones; Automated Trust Negotiation; DARPA Information Survivability Conference and Exposition; 2000

[BF99] D. Boneh and M. Franklin; An efficient public key traitor tracing scheme In Proceedings Crypto Lecture Notes in Computer Science Springer-Verlag; 1999; 338–353

[HBSO03] J. Holt and R. Bradshaw and K. E. Seamons and H. Orman; Hidden credentials In Proc ACM Workshop on Privacy in the Electronic Society. ACM Press 2003

[H03] J Horwitz; A Survey Of Broadcast Encryption; http://math.scu.edu/ jhorwitz/pubs/ 2003

[FN94] A. Fiat and M. Naor Broadcast Encryption In Advances in Cryptology CRYPTO Springer-Verlag 1994; 480-491

[1] If in this step there is more than one sensitive credential, Bob will send a nested encryption with the least sensitive credential being the upper most encryption and the resource being the lower most.

[CFN94] B. Chor and A. Fiat and M.Naor; Tracing Traitors Advances in Cryptology - Crypto Lecture Notes in Computer Science 1994; Vol 839; 257-270

[NNL01] D.Naor and M. Naor and J. Lotspiech Revocation and Tracing Schemes for Stateless Receivers Advances in Cryptology: CRYPTO, 2001

A Threshold Scheme with Disenrollment (extended abstract)

Michael Beiter

Wilhelm-Schickard Institut für Informatik
Universität Tübingen
Sand 14, 72076 Tübingen
Germany
http://www-dm.informatik.uni-tuebingen.de
beiter@informatik.uni-tuebingen.de

Abstract

We address the question of adding and in particular removing participants from secret sharing schemes when shares of unaffected parties shall remain unchanged. We will introduce a new management model for dynamic secret sharing schemes that allows unlimited disenrollment of parties from secret sharing schemes without need for broadcasts. As an example, we give an implementation based on Shamirs threshold scheme. We will see that using this model not only the disenrollement capability is not limited to a predefined constant but also arbitrary changes on the access structure are possible.

Keywords. secret sharing, threshold schemes with disenrollment

1 Introduction

To protect a secret from abuse or to add redundancy for increased reliability, it is common practice to use secret sharing schemes, in particular threshold schemes as introduced by Shamir [12] and Blakley [2]. A threshold scheme protects a secret g among v participants so that i participants, $u \leq i \leq v$, can recreate g and less than u participants can't uniquely determine g.

In this paper, we will study the question of adding and in particular removing participants from secret sharing schemes when shares of unaffected parties shall remain unchanged. We will introduce a new management model for dynamic secret sharing schemes that allows unlimited disenrollment of parties from arbitrary secret sharing schemes without need for broadcasts. As an example, we give an implementation based on Shamirs threshold scheme.

Disenrollment is needed when a authorized party should no longer have access to protected datasets or when shares are lost or disclosed shares [1]. Threshold schemes with disenrollement capabilities were introduced by Blakley *et al.*[1], and have been discussed in several publications (*e.g.*[4] [8] [9] [6] [13] [3] [5] [10]).

2 A "combiner" management model for dynamic SSS

Although there exist schemes that don't (*i.e.*[7]) or don't have the need to (*i.e.*schemes that rely on public reconstruction), it is common practice to split the dealer in two instances: A "dealer" who constructs the scheme and as well adds and removes parties from the access structure and a "combiner" who reconstructs the secret and triggers a certain action.

In common secret sharing schemes, the combiner is an entity that is not identical with the dealer, say the authority that is totally aware of the protected secret and any other information necessary for constructing the scheme. However, there are usually no further statements who the combiner is and what his abilities are. So in practice, the combiner can be one participant

or even all of them if broadcasts are used. In [11], Martin reviews various management models for secret sharing schemes with dynamic access structures.

In our model, which we demonstrate only for threshold schemes, the combiner is neither one of the shareholders, nor the dealer: We introduce the combiner as an additional trusted entity that has a secure link to the dealer and can store a certain amount of (secret) information. As the combiner relys upon the privacy of this secret information and the secure connection to the dealer to disenroll participants, we concede that the capabilitiy for disenrollment may be lost when these premises are no longer existent.

However, using his secret information and some shares provided by the participants, the combiner can reconstruct the secret if and only if the provided shares represent an authorized subset of the access structure. In other words, when the combiner's information is revealed, the scheme remains secure insofar as for recreating the secret, collaboration of an sufficient number of participants is required. As long as his information remains private, reconstruction of the secret is only possible upon approval by the combiner.

According to Martin's terminology [11], in the initialization phase our model has need for a dealer with secure connection both to the participants and the combiner.

In the running phase, the dealer remains present but does not have any connection to the participants. The secure connection to the combiner remains intact. In practice, this is much easier than keeping secure connections to all participants. The combiner in turn can either provide broadcast connections to the participants, what results in a one time secret sharing scheme, or use a secure channel to each party participating in the recovering process. As the participant's shares and the secret remain private, the latter situation allows using the scheme multiple times.

Connections between the participants are neither necessary in the initialization nor in the running phase.

As we will see in the following example, we can implement Martins type "A" techniques where a full dynamic sampling of access structures is possible.

3 A threshold scheme with disenrollment capability

3.1 Construction

Let $\mathbb{Z}_p$ be a finite field of prime order p, $g \in \mathbb{Z}_p$ be the secret and k_i the share of party $i, 1 \leq i \leq v$. Using a (u, v) threshold access structure, we create a polynomial f of degree v instead of $u-1$ as Shamir does in [12].

The dealer determines pairs (x_i, y_i) so that $x_0 = 0, y_0 = g$ and $k_i = (x_i, y_i), 1 \leq i \leq v$, holds. The y_i are chosen randomly in $\mathbb{Z}_p$, the x_i are pairwise distinct. Afterwards, he interpolates these $v+1$ points by a polynomial f using the Newton method. In the unlikely case that $deg(f) \neq v$, the dealer restarts the procedure.

When $deg(f) = v$ applies, the dealer computes $v-u+1$ pairs $(x_j, f(x_j)), -(v-u+1) \leq j \leq -1$, $x_j \neq x_0, x_1, \ldots x_v$, and passes them to the combiner. The shares k_i are distributed to the participants via a protected channel.

For numerical efficiency, the dealer doesn't store the g and the k_i but keeps a copy of the interpolated Newton polynomial for further use: Using Newton instead of Lagrange interpolation allows the dealer to reuse already computed results as intermediate results when adding new participants and is therefore more handy regarding efficiency aspects (see 3.3 for details).

3.2 Reconstruction

The reconstruction of the shared secret is only possible with assistance of the combiner: No authorized subset of u participants or more has enough shares to interpolate f and hence to determine g. Depending on the number of parties initiating the reconstruction process, the combiner will contribute 1 up to $v-u+1$ shares, interpolate the polynomial f and computes $f(0)=g$. This has the advantage that the combiner can check that no wildcat reconstructions occur, what assures that g is kept secret in any circumstances.

Since the combiner contributes up to $v-u+1$ shares, some steps can be precomputed to calculate the Newton polynomial: After the combiner received the additional shares $(x_j, f(x_j))$ from the dealer, he precomputes the first $v-u+1$ steps of the Newton polynomial and caches this as an intermediate result. Using the precomputed part of the Newton polynomial, the combiner can compute the polyomial using u shares and check any remaining parties by insertion as he could do in Shamirs scheme. So, utilization of the Newton instead of Lagrange polyonmial is more efficient.

3.3 Adding new particpants

Adding one or more new parties is straight forward: As v increases by n, the dealer needs to recompute the polynomial f, as its degree must also increase by n.

The dealer chooses n pairs $(x_i, y_i), v+1 \leq i \leq v+n$, where y_i is randomly in $\mathbb{Z}_p$, the x_i are pairwise disjoint and $x_i \neq 0, x_1, \ldots x_v$. Using the stored Newton polynomial, the dealer computes a new Newton polynomial f^* using the additional shares. This does not affect any of the existing shares k_i or the secret g. As stated above, the dealer must choose different values for the new y_i if $deg(f^*) \neq v+n$.

As in the construction step above, the dealer computes $v+n-u+1$ pairs $(x_j, f^*(x_j)), -(v+n-u+1) \leq j \leq -1$, $x_j \neq x_0, x_1, \ldots x_{v+n}$, and passes them to the combiner. The n created shares k_i are distributed to the new participants via a protected channel, the Newton polynomial f^* is stored by the dealer.

3.4 Disenrollment of participants

To remove a participant k from all authorized subsets, it must be assured that his share (x_k, y_k) is no longer part of f. For this, we create a new polynomial f^* that has all points $(x_i, y_i), 0 \leq i \leq v$, in common with f except the pair (x_k, y_k). To achieve this, the dealer chooses a y_k^* randomly in $\mathbb{Z}_p$ and interpolates the Newton polynomial f^* with the new (x_k, y_k^*) instead of (x_k, y_k).

Since the disenrolled user is not removed from the scheme but only deactivated, the degree of the polynomial remains v. So the dealer checks for $deg(f^*) = v$, computes $v-u+1$ pairs $(x_j, f^*(x_j)), -(v-u+1) \leq j \leq -1$, that replace the corresponding former pairs $(x_j, f(x_j))$ and passes them to the combiner. He doesn't have to inform the parties of the disenrollment, nor is there any need to communicate with the removed participant: The shares k_i of the remaining participants are unchanged, and the share $k_k = (x_k, y_k)$ becomes invalid after the combiner has updated his data.

The dealer can disenroll multiple parties by chosing multiple new polynomial values for the corresponding shares. It's easy to see that using this combiner driven management model, there are no limitations regarding any maximum disenrollment capacity or even further any changes in u or v. In general, we can say that using an adequate encoding for the shares, we could even replace the threshold access structure with an arbitrary one.

In the lifetime of a secret sharing scheme, the dealer should touch the share of each party only on creation and revocation: If a participant is disenrolled multiple times, there is a small chance that the random chosen pair (x_k, y_k^*) matches the original (x_k, y_k) and reactivates the share. To circumvent this, the dealer must keep records of deactivated parties.

4 Conclusions

In this paper, we addressed the problem of disenrolling one or more participants from a threshold scheme whereas shares of unaffected parties shall remain unchanged. We presented a new management model for dynamic secret sharing schemes allowing unlimited modification of threshold access structures without need for broadcasts. As an example, we gave a modification of the well known Shamir threshold scheme that allows adding and removing parties, ensuring that reconstruction as well as addition and removal are suspects to trusted authority control, whereas the scheme remains perfect.

We mention that this construction can be generalized for more general access structures.

References

[1] Blakley, B., Blakley G.R., Chan A.H., Massey J.L.: Threshold schemes with disenrollment, in "Advences in Cryptology - Crypto '92", *Lecture Notes in Computer Science*, Vol. 740, 540-548, Springer Verlag 1993

[2] Blakley, G.R.: Safeguarding Cryptographic Keys, in "Proceesings AFIPS 1979", *National Computer Conference*, Vol. 48, 313-317, 1979

[3] Blundo, C.: A note on dynamic threshold schemes, *Information Processing Letters*, Vol. 55, 189-193, 1995

[4] Blundo, C., Cresti, A., De Santis, A., Vaccaro, U.: Fully dynamic secret sharing schemes, in "Advences in Cryptology - Crypto '93", *Lecture Notes in Computer Science*, Vol. 773, 110-125, Springer Verlag 1994

[5] Cachin, C.: On-Line Secret Sharing, in "Proceedings of the 5th IMA Conference on Cryptography and Coding", *Lecture Notes in Computer Science*, Vol. 1025, 190-198, Springer Verlag 1995

[6] Charnes, C., Pieprzyk J., Safavi-Naini R.: Conditionally secure secret sharing schemes with disenrollment capability, *Proceedings of the 2nd ACM Conference on Computer and communications security*, 89-95, 1994

[7] Ghodosi, H., Pieprzyk J.,Safavi-Naini, R.: Dynamic Threshold Cryptosystems: A New Scheme in Group Oriented Cryptography, in "Proceedings of PRAGOCRYPT '96", *International Conference on the Theory and Applications of Cryptology*, 370-379, 1996

[8] Li, M., Poovendran, R.: A Note on Threshold Schemes with Disenrollment, *Conference on Information Sciences and Systems*, The Johns Hopkins University, March 12-14, 2003

[9] Li, M., Poovendran, R.: Broadcast Enforced Threshold Schemes with Disenrollment, in "Selected Areas in Cryptography", *Lecture Notes in Computer Science*, Vol. 3006, 101-116, Springer Verlag 2004

[10] Martin, K.M: Untrustworty participants in perfect secret sharing schemes, *Cryptography and Coding III*, 255-264, Oxford University Press, 1993

[11] Martin, K.M.: Dynamic access policies for unconditionally secure secret sharing schemes, *Proceedings of IEEE Information Theory Workshop (ITW 05)*, Awaji Island, Japan 2005

[12] Shamir, A.: How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, 612-613, November 1979

[13] Sun, H.-U., Shieh, S.-P.: On dynamic threshold schemes, *Information Processing Letters*, Vol. 52, 201-206, 1994

A New Threshold Audio Secret Sharing Scheme[1]

Mohammad Ehdaie*, Taraneh Eghlidos**, Mohammad Reza Aref***

*Information System and Security Lab. (ISSL), School of Electrical Engineering,
Sharif University of Technology, Tehran, Iran.
mohammad@ehdaie.com

**Electronics Research Center,
Sharif University of Technology, Tehran, Iran.
teghlidos@sharif.edu

***Information System and Security Lab. (ISSL), School of Electrical Engineering,
Sharif University of Technology, Tehran, Iran.
aref@sharif.edu

Abstract

In this paper, a new audio secret sharing scheme is proposed. Where the previous schemes were $(2, n)$, this scheme is (k, n) threshold for $k \geq 2$, which is proven to be perfect and ideal. It is assumed that both share and secret are audio files instead of a bit string secret proposed in the previous works. The audio secret is reconstructed without any computation, that is only by playing audio shares simultaneously.

Keywords. Threshold Secret Sharing Scheme, Audio Secret Sharing Scheme.

1 Introduction

Secret sharing is a method to distribute a secret between some participants such that particular subsets (i.e. authorized subsets) could obtain the secret, whereas unauthorized subsets could not. In a perfect scheme, unauthorized subsets could not get any information about the secret by putting their shares together. If the cardinality of all minimum allowed subsets, k, is constant, then we have a threshold (k, n) secret sharing scheme, where n is the number of participants. In a perfect (k, n) threshold secret sharing scheme we should have:

i) Every k or more participants can obtain the secret by pooling their shares together.

ii) Every $k-1$ or fewer participants cannot obtain any information about the secret by pooling their shares together.

Secret sharing was first introduced by Blakley [Bl79] and Shamir [Sh79] in 1979, independently. In 1994, Naor and Shamir proposed a new (k, n) threshold visual secret sharing scheme [NS94], which was perfect and ideal. Ideal in the sense that the size of each share is equal to the size of the secret. The visual secret is reconstructed without any computation. In 1998, Desmedt et al. proposed a $(2, n)$ threshold Audio Secret Sharing Scheme (ASSS) which is perfect [DHQ98]. In their scheme, shares are audio files but the secret is a bit string, which is reconstructed without any extra computation.

Lin et al. in 2003 proposed a new ASSS which in addition to all above properties is ideal [LLY03].

Sometimes, the secret is not a bit string or visual and we deal with audio secrets. In this cases, it could be desired to reconstruct the audio secret without any computaion.

[1]This work was partially supported by ITRC.

In this paper, a new ASSS with audio secret is proposed, aiming at reconstructing the secret without any computation. Our scheme is a perfect and an ideal (k, n) threshold scheme.

This paper is organized as follows: In section 2, some properties of human audio system and audio files are given and used in our scheme. Section 3 deals with describing our scheme, proving that it is perfect and ideal. Section 4 covers all discutions and results. Finally, Section 5 summerizes the paper and gives the concluding remarks.

2 Preliminaries

Assume that we divide the audio file F into some small intervals and refer to the amplitude of the audio signal in the i^{th} interval as A_i. (for example 'wavread' function in MATLAB software takes a wave file and gets an array of size $s \times f$, where s is the file size (in seconds) and f is the sampling frequency (in Hz.)). The value of the i^{th} entry in the array is equal to the amplitude of the wave file in time i/f. As we know, human audio system is not sensitive to the phase of the signal. So, if we multiply all A_i values by -1 (i.e. a 180 degree phase shift) and replay them, we can hear the same sound.

Multiplying all A_i values by a constant number c, leads to amplifying the audio file with the coefficient c (i.e. changing the volume of the main sound).

Assume that we have two audio files of the same size whose amplitude vectors are A and B respectively. By playing both files simultaneously in the same place, we can hear a sound whose amplitude vector is $A + B$ (i.e. the interference property of the two sounds).

Based on the above facts, if we have a main audio file with the amplitude vector A and generate B_i and C_i such that $\forall i : B_i + C_i = aA_i$ where 'a' is a constant number and A_i is the i^{th} entry of A, by playing two audio files B and C, we can hear the main sound A which is amplified with coefficient 'a' .

In the rest of this paper, we consider one interval of audio files and share the secret A, as the amplitude of the sound in this interval. We mean by S_j , the amplitude of j^{th} share file in the same interval. This process should be done for all intervals. After generating S_j for all intervals, we put all S_{ji}'s together to obtain an audio file as j^{th} share file. Note that all coefficients are constants in each interval. Otherwise, we have distortion in audio signal and we cannot hear the main sound.

3 The New (k, n) Threshold ASSS

Assume we need to share an audio secret between n participants. In this case, we should divide the audio secret file into some small intervals and generate the shares in each interval independently. Then we put all share parts of each participant together and generate an audio share. Let A be the amplitude of the secret file in a particular interval and $S_1, S_2, \ldots, S_n$ be the amplitudes of the n share files in the same interval.

Now, we generate the share samples $S_1, S_2, \ldots, S_n$, such that each k participants could obtain A or a constant multiple of A and every $k-1$ or fewer participants could not get any information about A.

3.1 Shares Generation

1) Generate $S_1, S_2, \ldots, S_{k-1}$, randomly.
2) For $k \leq i \leq n$, compute S_i from the equation below:

$$e_i : \quad b_{i,1}S_1 + b_{i,2}S_2 + \ldots + b_{i,k-1}S_{k-1} + S_i = a_iA \tag{1}$$

Where a_i values are public or private nonzero constants and $b_{i,j}$ values are such that for every i_1, i_2, j_1, j_2, $k \leq i_1, i_2 \leq n$, $1 \leq j_1, j_2 \leq k-1$, the 2-dimensional vectors $(b_{i1,j1}, b_{i1,j2})$ and $(b_{i2,j1}, b_{i2,j2})$ are linearly independent (i.e. $b_{i1,j1}b_{i2,j2} \neq b_{i1,j2}b_{i2,j1}$). We can reach this condition by selecting $b_{i,j} = i^{j-1}$. Note that $b_{i,j}$ values could be public or private.

3.2 Secret Reconstruction

Lemma 3.1 *Assume $P_{j_1}, P_{j_2}, \ldots, P_{j_k}$ are k participants who want to reconstruct the secret. There is a linear combination of equations $e_k, e_{k+1}, \ldots, e_n$ leading to an equation of the form below:*

$$\beta_{j_1} S_{j_1} + \beta_{j_2} S_{j_2} + \ldots + \beta_{j_k} S_{j_k} = a'A$$

Proof: For simplicity, let $j_1, j_2, \ldots, j_k$ be a subset of $k, k+1, \cdots, n$. The proof of general case is straightforward.
We multiply both sides of the equation e_{j_i}, for $1 \leq i \leq k$, by α_{j_i} and then add the corresponding sides together. So we have:

$$\sum_{i=1}^{k} \alpha_{j_i} e_{j_i} : \qquad \sum_{i=1}^{k-1} \beta_i S_i + \sum_{i=1}^{k} \beta_{j_i} S_{j_i} = a'A$$

Where:

$$a' = \sum_{i=1}^{k} \alpha_{j_i} a_{j_i}$$

$$\beta_t = \sum_{i=1}^{k} \alpha_{j_i} b_{j_i,t} \quad 1 \leq t \leq k-1$$

$$\beta_{j_t} = \alpha_{j_t} \quad 1 \leq t \leq k$$

We are going to find $\alpha_{j_1}, \alpha_{j_2}, \ldots, \alpha_{j_k}$, such that β_{j_t} values for $1 \leq t \leq k$ are nonzero and β_t values, $1 \leq t \leq k-1$, are equal to zero. Consider the system of linear equations below:

$$\begin{cases} b_{j_1,1}\alpha_{j_1} + b_{j_2,1}\alpha_{j_2} + \ldots + b_{j_k,1}\alpha_{j_k} = 0 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\quad \vdots \\ b_{j_1,k-1}\alpha_{j_1} + b_{j_2,k-1}\alpha_{j_2} + \ldots + b_{j_k,k-1}\alpha_{j_k} = 0 \end{cases} \tag{2}$$

We have $k-1$ linearly independent equations and k unknowns. Hence, the above system has infinite answers. Let one of the unknowns be zero. Thus, we have $k-1$ linearly independent equations and $k-1$ unknowns. So, all α_{j_t}, $1 \leq t \leq k-1$, must be zero. As the system (2) has infinite answers, therefore, it has an answer such that all α_{j_t} values are nonzero. Hence, there is a linear combination of the equations $e_{j_1}, e_{j_1}, \ldots, e_{j_k}$, such that all β_t values, $1 \leq t \leq k-1$, are equal to zero and all β_{j_t} values, $1 \leq t \leq k$, are nonzero. Therefore, $P_{j_1}, P_{j_2}, \ldots, P_{j_k}$ can obtain the secret by replaying their shares, each with a constant coeffitient. (i.e. changing the volume of his/her audio share.)

Note that in (1), a_i values must not lead to $a' = 0$. Selecting such a_i values is easy.

3.3 Security

Lemma 3.2 *There is no linear combination of equations $e_k, e_{k+1}, \ldots, e_n$, such that the coefficients of only $k-1$ or fewer shares are nonzero.*

Proof: First, we consider a linear combination of two equations e_k and e_{k+1}:

$$\alpha_k e_k + \alpha_{k+1} e_{k+1} : \sum_{i=1}^{k-1} \beta_i S_i + \alpha_k S_k + \alpha_{k+1} S_{k+1} = a'A \tag{3}$$

If α_k and α_{k+1} are zero, then the above equation leads to no information. If one of them is zero, then the left side has k shares. If both of them are nonzero, then the left side contains S_k, S_{k+1} and at least $k-2$ shares from $S_1, S_2, \ldots, S_{k-1}$. If the left side contains only $k-3$ or fewer shares from $S_1, S_2, \ldots, S_{k-1}$, it means that at least two β values are zero and it is impossible. Assume β_i and β_j have zero values. So, we have:

$$\beta_i = 0 \longrightarrow b_{k,i}\alpha_k + b_{k+1,i}\alpha_{k+1} = 0$$

$$\beta_j = 0 \longrightarrow b_{k,j}\alpha_k + b_{k+1,j}\alpha_{k+1} = 0$$

The above equations have a nonzero answer iff the determinant of coefficients is zero (i.e. $b_{k,i}b_{k+1,j} = b_{k,j}b_{k+1,i}$). We assumed in section 3.1, b values are such that the above determinant is always nonzero. Hence, the linear combination of two equations has at least k shares in the left side.

Now, we can state a linear combination of equations $e_k, e_{k+1}, \ldots, e_n$ of the form below:

$$(\ldots(((\alpha_k e_k + \alpha_{k+1} e_{k+1}) + \alpha_{k+2} e_{k+2}) + \alpha_{k+3} e_{k+3})\ldots) + \alpha_n e_n$$

In each stage, a new share appears and at most one share is eliminated, because, in each stage, the coefficient of the share S_i, $1 \leq i \leq k-1$, is multiplied by a constant coefficient and then added to a multiple of the coefficient of another share. Therefore, the nonzero-determinant-condition is still valid. Hence, every linear combination of the equations $e_k, e_{k+1}, \ldots, e_n$, contains at least k shares in the left hand side.

3.4 Ideality and Practical Issues

If the secret is an audio file of size s (in seconds), then the size of each share would be s too. Hence, this scheme is ideal.

When we would like to use this scheme as an ASSS, we have to consider some practical issues, such as physical properties of amplifiers and so on, which are out of the scope of this paper.

4 Discutions and Results

This scheme is perfect and ideal. Besides, the secret was reconstructed without any computation, like previous ones. In adition, our scheme has some extra features. It is a general (k, n) threshold scheme and uses an audio secret instead of a bit string.

5 Conclusion

In this paper, we proposed a new audio secret sharing scheme which is proven to be perfect and ideal. Our scheme is a (k, n) threshold scheme, instead of $(2, n)$ schemes which have already been proposed. Shares are audio files, so is the secret. Audio secret is obtained only by playing k shares which are amplified by a specific coefficient. Therefore, the secret is reconstructed without any computation.

References

[Bl79] Blakley G. R., *"Safeguarding cryptographic keys"*, in Proceedings of the National Computer Conference, 1979, American Federation of Information Processing Societies Proceedings 48, pp. 313-317, (1979).

[Sh79] Shamir A., *"How to share a secret"*, Commun. Of the ACM, 22 (11), pp. 612-613, (1979).

[NS94] Naor M., Shamir A., *"Visual cryptography"*, Eurocrypt 94, pp. 1-12, (1994).

[DHQ98] Desmedt Y., Hou S., Quisquater J., *"Audio and optical cryptography"*, in Advances in Cryptology-Asiacrypt '98, Springer-Verlag LNCS, pp. 392-404, (1998).

[LLY03] Lin C. C., Laih C. S., Yang C. N., *"New Audio Secret Sharing Schemes With Time Division Technique"*, J. of Information Science and Engineering, 19, pp. 605-614 (2003).

Attacks on the Stream Ciphers TPy6 and Py6 and Design of New Ciphers TPy6-A and TPy6-B

Gautham Sekar, Souradyuti Paul and Bart Preneel

Katholieke Universiteit Leuven, Dept. ESAT/COSIC,
Kasteelpark Arenberg 10,
B–3001, Leuven-Heverlee, Belgium
{gautham.sekar,souradyuti.paul,bart.preneel}@esat.kuleuven.be

Abstract

The stream ciphers Py, Pypy and Py6 were designed by Biham and Seberry for the ECRYPT-eSTREAM project in 2005. The ciphers were promoted to the 'Focus' ciphers of the Phase II of the eSTREAM project. Unfortunately, due to discovery of some cryptanalytic weaknesses, strengthened versions of the ciphers, namely TPy, TPypy and TPy6, were built. So far there exists no attacks on TPy6. In this paper, we find hitherto unknown weaknesses in the keystream generation algorithms of the Py6 and of its stronger variant TPy6. Exploiting these weaknesses, a large number of distinguishing attacks are mounted on the ciphers, the best of which works with 2^{233} data and comparable time. In the second part, we discover existence of weak related keys of Py6 and TPy6. Finally, we present two new ciphers derived from the TPy6, namely TPy6-A and TPy6-B, whose performances are 2.65 cycles/byte and 4.4 cycles/byte on Pentium III. As a result, to the best of our knowledge, TPy6-A becomes the fastest stream cipher in the literature. Based on our security analysis, we conjecture that no attacks lower than brute force are possible on the ciphers TPy6-A and TPy6-B.

Keywords. Stream Ciphers, Array, Modular Addition, Py

1 Introduction

Timeline: the Py-family of Ciphers

- **April 2005.** The ciphers Py and Py6, designed by Biham and Seberry, were submitted to the ECRYPT project for analysis and evaluation in the category of software based stream ciphers [4]. The impressive speed of the cipher Py in software (about 2.5 times faster than the RC4) made it one of the fastest and most attractive contestants.
- **March 2006 (at FSE 2006).** Paul, Preneel and Sekar reported distinguishing attacks with $2^{89.2}$ data and comparable time against the cipher Py [15]. Crowley [7] later reduced the complexity to 2^{72} by employing a Hidden Markov Model.
- **March 2006 (at the Rump session of FSE 2006).** A new cipher, namely Pypy, was proposed by the designers to rule out the aforementioned distinguishing attacks on Py [5].
- **May 2006 (presented at Asiacrypt 2006).** Distinguishing attacks were reported against Py6 with 2^{68} data and comparable time by Paul and Preneel [16].
- **October 2006 (to be presented at Eurocrypt 2007).** Wu and Preneel showed key recovery attacks against the ciphers Py, Pypy, Py6 with chosen IVs. This attack was subsequently improved by Isobe *et al.* [9].

- **January 2007.** Three new ciphers TPypy, TPy, TPy6 were proposed by the designers [3]; the ciphers can very well be viewed as the strengthened versions of the previous ciphers Py, Pypy and Py6 where the above attacks do not apply.
- **February 2007.** Sekar *et al.* published attacks on TPy and TPypy, each of which required 2^{281} data and comparable time [17]. So far there exist no attacks on the cipher TPy6.

2 Distinguishing attacks on the Py6 and the TPy6

We detect a large number of input-output correlations of TPy6 and Py6 that allow us to build distinguishers. One such correlation is outlined in Theorem 2.1. In the following theorem $O_{i(j)}$ denotes the j-th bit of the first outputword generated at round i and $P_n[X]$ denotes the element X of array P generated in the n-th round.

Theorem 2.1 $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = 0$ *if the following* 17 *conditions are simultaneously satisfied.*

1. $P_1[26] \equiv -18 \bmod 32$,
2. $P_2[26] \equiv 7 \bmod 32$,
3. $P_3[26] \equiv -4 \bmod 32$,
4. $P_7[26] \equiv 3 \bmod 32$,
5. $P_8[26] \equiv 3 \bmod 32$,
6. $P_1[18] = P_2[57] + 1$,
7. $P_1[57] = P_2[18] + 1$,
8. $P_7[18] = P_8[18] + 1$,
9. $P_7[57] = P_8[57] + 1$,
10. $P_3[18] = 62$,
11. $P_1[8] = P_3[57] + 2$,
12. $P_1[18] = 3$,
13. $P_3[8] = 0$,
14. $P_1[57] = P_7[8] + 6$,
15. $P_7[48] = 60$,
16. $P_6[48] = P_8[8] + 2$,
17. $d_{7(i-7)} \oplus d_{8(i-7)} \oplus c_{1(i)} \oplus d_{3(i)} \oplus d_{1(i+7)} \oplus c_{3(i+7)} \oplus c_{7(i+7)} \oplus e_{7(i+7)} \oplus c_{8(i+7)} \oplus e_{8(i+7)} = 0$.[1]

The distinguisher, constructed from Theorem 2.1, works with 2^{233} randomly chosen key/IVs and as many outputwords (each key generating one outputword). In the design specifications, the TPy6 and the Py6 are claimed to be compatible with key size ranging from 8 bits to 2048 bits. The results testify to the fact that the security of TPy6 (and Py6 also) does not grow exponentially with the key size because, if the ciphers are used with key size longer than 233 bits, our attacks are better than exhaustive search. It is also worth noting that the distinguisher can be built within the design specifications of the ciphers. To derive the distinguisher, the adversary chooses 2^{233} randomly chosen key/IVs and for each of them she collects one outputword. Note that, according to the design specification, the ciphers can run for 2^{61} rounds (since each round generates 8 bytes as output) per key where our best distinguisher requires only 8 rounds per key. Hence the TPy6 is theoretically broken under the design specifications when the size of the key is greater than 29 bytes. In addition to the above distinguisher, we detect biases in a large number of outputs at rounds r, $r+2$, t and u where $r > 0$; $t, u \geq 5$; $t \notin \{r, r+2, u\}$; $u \notin \{r, r+2, t\}$. We provide a general framework to compute the biases due to the presence of arbitrarily many weak states. However, we were unable to combine those biases into a more efficient attack. Combining multiple distinguishers into a single and more efficient one is still an alluring open problem. The details of the attacks will be provided in the extended version of the paper.

[1] The terms c, d, e are the carries generated in certain expressions, the descriptions of which can be found in the proof of Theorem 2.1 which will be provided in the final version.

3 Related-key attacks on the Py6 and the TPy6

In a related key attack, the attacker chooses a relation f between a pair of keys key_1 and key_2 (e.g., $key_1 = f(key_2)$) rather than the actual values of the keys and extracts secret information from a cryptosystem using the relation f [2, 11]. Related-key weakness is a cause for concern in a protocol where key-integrity is not guaranteed or when the keys are generated manually rather than from a pseudorandom number generator [10]. Furthermore, there is a growing tendency by the designers nowadays to build hash functions from stream ciphers [6] instead of building them from block ciphers. In such attempts, related-key weaknesses of stream ciphers should be carefully analyzed and removed from the designs. In this paper, it is shown that the key setup algorithm of the TPy6 (and the Py6) has weaknesses. Exploiting these weaknesses, we construct distinguishing attacks on the ciphers, where the best distinguisher is built when the cipher used with 64-byte long keys.

Given two keys, key_1 and key_2 (each key is 64 bytes long) and identical 16 byte long IVs, such that,

1. $key_1[16] \oplus key_2[16] = 1$,
2. the lsb of $key_1[16]$ is 1,
3. $key_1[17] \neq key_2[17]$ and
4. $key_1[i] = key_2[i]\ \forall i \notin \{16, 17\}$.

the outputs at rounds 1 and 3 are found to be correlated. The data complexity of the distinguisher hence constructed is calculated to be $2^{172.8}$. These related-key attacks work with any IV-size ranging from 16 bytes to 64 bytes. However, the attack complexities increase with shorter keys. Note that the usage of long keys in the Py-family of ciphers makes it very attractive to be used as fast hash functions (e.g., by replacing the key with the message). In such cases, these related-key weaknesses can turn out to be serious impediments. Elaborate discussion of the attacks will be included in the final version.

4 Two New Ciphers: TPy6-A and TPy6-B

The Py-family of stream ciphers have been subject to extensive analysis ever since they were proposed in April 2005. The impressive speeds of the ciphers in software, particularly the Py6 and the TPy6, have motivated us to modify the TPy6 to rule out all the attacks described in the previous sections of the paper. Firstly, many attacks on the Py, in particular, [15], [7], [21] and [9] can be easily translated to attacks on the Py6. However, due to smaller internal state of the TPy6, the attack described in [17] does not apply to TPy6. Secondly, the speed of execution of Py6 on Pentium-III is about 2.82 cycles/byte which is very fast. These observations make TPy6 and Py6 more favorable to be used as fast stream ciphers than Py. The TPy6 is resistant to the attacks described in [21], [9]. In order to generate a fast and secure stream cipher, we redesign the TPy6 where the variable rotation of a 32-bit term s in the round function is replaced by a constant, non-zero rotation term. We choose 19 for the constant rotation,

however, we believe that any non-zero rotation between 1 and 31 is also safe (but a proof is non-trivial). The resultant cipher is named TPy6-A. It is shown that this tweak clearly reduces one addition operation in each round (thereby, the performance is improved) and makes the cipher secure against all the existing attacks on Py6 and TPy6. A relatively slower version, where one outputword is removed from each round of TPy6, is also proposed. The speeds of execution of the TPy6-A and TPy6-B on Pentium-III are 2.65 cycles/byte and 4.4 cycles/byte. Our security analysis conjectures that the TPy6-A and TPy6-B are immune to all attacks lower than brute force. Algorithm 1 describes the PRBGs of the TPy6-A and TPy6-B. Note that the key/IV setup algorithms of the ciphers are identical to the key/IV setup of TPy6.

Algorithm 1 A Step of TPy6-A and TPy6-B

Require: $Y[-3, ..., 64]$, $P[0, ..., 63]$, a 32-bit variable s
Ensure: 64-bit random (TPy6-A) or 32-bit random (TPy6-B) output
/*Update and rotate P*/
1: swap ($P[0]$, $P[Y[43]\&63]$);
2: rotate (P);
/* Update s*/
3: $s+ = Y[P[18]] - Y[P[57]]$;
4: $s = ROTL32(s, 19)$; /***Tweak:** variable rotation in TPy6 replaced by a *constant non-zero* rotation*/
/* Output 8 bytes (least significant byte first)*/
5: output $((ROTL32(s, 25) \oplus Y[64]) + Y[P[8]])$;/*this step is removed for TPy6-B*/
6: output $((\quad s \quad \oplus Y[-1]) + Y[P[21]])$;
/* Update and rotate Y*/
7: $Y[-3] = (ROTL32(s, 14) \oplus Y[-3]) + Y[P[48]]$;
8: rotate(Y);

5 Conclusions and Open Problems

The first contribution of the paper is the development of a family of distinguishers from the outputs at rounds r, $r + 2$, t and u of the cipher TPy6 (and Py6), where $r > 0$; $t, u \geq 5$; $t \notin \{r, r + 2, u\}$; $u \notin \{r, r + 2, t\}$. The best distinguisher works with data complexity 2^{233}. It is reasonable to assume that these weak states can be combined to mount a more efficient attack on TPy; however, methods to combine many distinguishers into a single yet more efficient one is still an open problem. The second contribution of the paper is the detection of weak related-keys in TPy6 and Py6. Exploiting the weaknesses in the key setup algorithm of the TPy6, a distinguisher with data complexity $2^{172.8}$ is built on the cipher. The final contribution is a proposal of two new, extremely fast stream ciphers TPy6-A and TPy6-B, which rule out all the existing attacks on the TPy6 and are conjectured to be immune to all attacks lower than brute force.

References

[1] T. Baignères, P. Junod and S. Vaudenay, "How Far Can We Go Beyond Linear Cryptanalysis?," *Asiacrypt 2004* (P. Lee, ed.), vol. 3329 of *LNCS*, pp. 432–450, Springer-Verlag, 2004.

[2] E. Biham, "New Types of Cryptanalytic Attacks Using Related Keys," J. Cryptology, vol 7(4), pp. 229-246, 1994.

[3] E. Biham, J. Seberry, "Tweaking the IV Setup of the Py Family of Ciphers – The Ciphers Tpy, TPypy, and TPy6," Published on the author's webpage at `http://www.cs.technion.ac.il/ biham/`, January 25, 2007.

[4] E. Biham, J. Seberry, "Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays," *ecrypt submission*, 2005.

[5] E. Biham, J. Seberry, "Pypy (Roopy): Another Version of Py," *ecrypt submission*, 2006.

[6] D. Chang, K. Gupta, M. Nandi, "RC4-Hash : A New Hash Function based on RC4 (Extended Abstract)," Indocrypt 2006 (R. Barua, T. Lange, eds.), vol 4329 of LNCS, Springer-Verlag, 2006.

[7] P. Crowley, "Improved Cryptanalysis of Py," *Workshop Record of SASC 2006 - Stream Ciphers Revisited*, ECRYPT Network of Excellence in Cryptology, February 2006, Leuven (Belgium), pp. 52-60.

[8] S. Fluhrer, I. Mantin, A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," Selected Areas in Cryptography 2001 (S. Vaudenay, A. Youssef, eds.), vol 2259 of LNCS, pp. 1-24, Springer-Verlag, 2001.

[9] T. Isobe, T. Ohigashi, H. Kuwakado M. Morii, "How to Break Py and Pypy by a Chosen-IV Attack," eSTREAM, ECRYPT Stream Cipher Project, Report 2006/060.

[10] J. Kelsey, B. Schneier, D. Wagner, "Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA," ICICS 1997 (Y. Han, T. Okamoto, S. Qing, eds.), vol 1334 of LNCS, pp 233-246, Springer-Verlag, 1997.

[11] J. Kelsey, B. Schneier, D. Wagner, "Key-Schedule Cryptoanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES," CRYPTO 1996 (N. Koblitz, ed.), vol. 1109 of LNCS, pp. 237-251, Springer-Verlag, 1996.

[12] L. R. Knudsen, "Cryptanalysis of LOKI," Advances in Cryptology ASIACRYPT '91 (H. Imai, R. Rivest, T. Matsumoto), vol 739 of LNCS, Springer-Verlag, 1993, pp. 22-35.

[13] L.R. Knudsen, "Cryptanalysis of LOKI91," Advances in Cryptology AUSCRYPT '92, Springer-Verlag, 1993, pp. 196-208.

[14] O. Dunkelman, E. Biham, N. Keller, "A Simple Related-Key Attack on the Full SHACAL-1," CT-RSA 2007 (Masayuki Abe, ed.), vol 4377 of LNCS, Springer-Verlag, 2007.

[15] S. Paul, B. Preneel, G. Sekar, "Distinguishing Attacks on the Stream Cipher Py," *Fast Software Encryption 2006* (M. Robshaw, ed.), vol. 4047 of *LNCS*, pp. 405-421, Springer-Verlag, 2006.

[16] S. Paul, B. Preneel "On the (In)security of Stream Ciphers Based on Arrays and Modular Addition," *Asiacrypt 2006* (X. Lai and K. Chen, eds.), vol. 4284 of *LNCS*, pp. 69-83, Springer-Verlag, 2006.

[17] G. Sekar, S. Paul, B. Preneel, "Weaknesses in the Pseudorandom Bit Generation Algorithms of the Stream Ciphers TPypy and TPy," available at `http://eprint.iacr.org/2007/075.pdf`.

[18] D. Wagner, "The Boomerang Attack," proceedings of Fast Software Encryption 2001, Lecture Notes in Computer Science 1636, pp. 156170, 1999.

[19] X. Wang, X. Lai, D. Feng, H. Chen, X. Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD," Advances in Cryptology, proceedings of EUROCRYPT 2005, Lecture Notes in Computer Science 3494, pp. 118, 2005.

[20] X. Wang, H. Yu, "How to Break MD5 and Other Hash Functions," Advances in Cryptology, proceedings of EUROCRYPT 2005, Lecture Notes in Computer Science 3494, pp. 1935, 2005.

[21] H. Wu, B. Preneel, "Differential Cryptanalysis of the Stream Ciphers Py, Py6 and Pypy," *Eurocrypt 2007* (to appear).

The Impact of Entropy Loss caused by Random Functions

Andrea Röck

Projet CODES, INRIA Rocquencourt, France
http://www-rocq.inria.fr/codes/
andrea.roeck@inria.fr

Abstract

We consider the problem of entropy loss in a stream cipher which uses a random function to update its inner state. For small numbers of iterations, we introduce a new approximation of the state entropy. This improves the known upper bound given by the number of image points. Subsequently, we discuss two collision attacks which are based on the entropy loss after multiple iterations. For both attacks, we provide a detailed analysis of the complexity.

Keywords. random functions, entropy, stream cipher

1 Introduction

Recently, several stream ciphers have been proposed with a non-bijective update function. Moreover, for several of those propositions, the update function is expected to behave like a random function. Using a random function instead of a random permutation induces an entropy loss in the state. An attacker might exploit this fact to mount an attack. We will examine the effectiveness of such an approach.

This article is divided in two main sections. In Section 2, we present a new entropy estimation which is, for small numbers of iterations, more precise than the previous results. In Section 3, we discuss two collision attacks against MICKEY version 1 [BD05] presented in [HK05]. For the first attack, we show that we only gain a factor on the time complexity by increasing the data complexity by the same factor. For the second attack, we demonstrate that, contrary to what is expected from the results in [HK05], the complexities are equivalent to a direct collision search in the starting values.

2 Estimation of Entropy

In this section we discuss the estimation of the inner state entropy and some useful properties of random functions.

DEFINITION 2.1 *Let $\mathcal{F}_n = \{\varphi \mid \varphi : \Omega_n \to \Omega_n\}$ be the set of all functions which map a set $\Omega_n = \{\omega_1, \omega_2, \ldots, \omega_n\}$ of n elements onto itself. If we choose φ randomly from $\mathcal{F}_n$, we refer to it as* random function *or* random mapping.

Let S be the internal state of the stream cipher with values in Ω_n and $\varphi \in \mathcal{F}_n$ the update function. By $(s_k)_{k\geq 0}$ we denote the value of the inner state after $k \geq 0$ iterations. We assume that the initial state s_0 and the function φ are chosen randomly and independently. In practice, they might be determined by the secret key and the IV. In each iteration φ is applied to the current state s_k to obtain the new state s_{k+1}, i.e. $s_{k+1} = \varphi(s_k)$. For a given φ, let $p_i^{\varphi}(k)$ be the probability to reach the state ω_i after k iterations for a fixed distribution of the initial value s_0. We then refer to

$$H_k^{\varphi} = \sum_{i=1}^{n} p_i^{\varphi}(k) \log_2 \left(\frac{1}{p_i^{\varphi}(k)} \right)$$

as the *entropy of the state* at time k, if φ was chosen as the update function. $E(H_k)$ represents the expected entropy after k iterations, where the expectation is taken uniformly over all $\varphi \in \mathcal{F}_n$.

2.1 Previous Works

Flajolet and Odlyzko provide, in [FO90], a wide range of properties of random functions by analyzing their functional graphs. Such a graph consists of separated components, where each component is build by a cycle of trees, i.e. the nodes in the cycle are the root of a tree. To find a specific property of random functions, Flajolet and Odlyzko construct the generating function of this property in the functional graph. Subsequently, they obtain the exponent for a given n, and thus, the desired property, by means of a singularity analysis of the generating function.

Some of the properties given in the article are the number of cycle point $cp(n) = \sqrt{\pi n/2}$, the maximal tail length $mtl(n) = \sqrt{\pi n/8}$, the number of r-nodes $rn(n,r) = n/r!e$ and the number of image points after k iterations $ip(n,k) = (1-\tau_k)\,n$ where $\tau_0 = 0$ and $\tau_{k+1} = e^{-1+\tau_k}$. By the maximal tail length we mean the maximal number of steps before reaching a cycle, an r-node is a node in the tree with exactly r incoming nodes and by the image points we mean all points in the graph that are reachable after k iterations of the function. These values represent always the expectation taken over all functions $\mathcal{F}_n$ and are approximations for n large enough.

Hong and Kim, in [HK05], used the number of image points to give an upper bound of the state entropy after k iterations of a random function. They utilized the fact that the entropy is always smaller or equal than the logarithm of the number of points with probability larger than zero. After a finite number of steps, each point in the functional graph will reach a cycle, thus the number of image points can never drop under the number of cycle points. Therefore, the upper bound of the estimated entropy

$$E(H_k) \le \log_2(n) + \log_2(1-\tau_k) \tag{1}$$

is valid only as long as $ip(n,k) > cp(n)$. In Figure 1, we compare the empirical gained state entropy for $n = 2^{16}$ and different numbers k of iterations, with the estimated number of image points, cycle points and the maximal tail length.

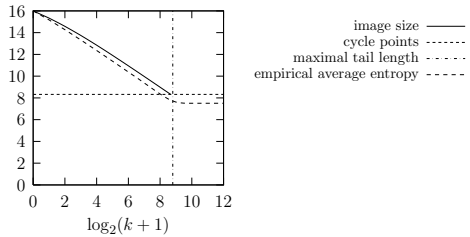


Figure 1: Empirical average entropy in comparison with theoretical upper bound for $n = 2^{16}$

For the empirical entropy we chose 10^4 random functions, by means of the HAVEGE random number generator [SS02], and calculated the average entropy under the assumption of a uniform distribution of the initial state. Even if n is not very big, it is sufficient to understand the relation between the different factors. We can see in the graph that if k stays smaller than $mtl(n)$ this bound stays valid and does not drop under $\log_2(cp(n))$.

2.2 New Entropy Estimation

The number of image points provides only an upper bound for the expected entropy. We tried to find a more precise approximation by employing the methods and properties stated in [FO90].

For a given function $\varphi \in \mathcal{F}_n$, let ω_i be a node with r incoming branches (r-node). The idea is that this is equivalent to the fact that ω_i is produced by exactly r different starting values after one iteration. Thus, if the initial distribution of the state is uniform, this node has the probability $p_i^{\varphi}(1) = r/n$.

For one iteration we can use directly the expected number of r nodes, $n/r!e$, given in [FO90]. For more than one iteration we need to find a new property. Let $n\,c_k(r)$ be the expected number of points that are reached by exactly r tree-nodes after k iterations, thus $c_k(r)$ denotes a fraction. Then in the case of a uniform initial distribution the expected entropy is

$$E(H_k) \approx \sum_{r=1}^{n} n\,c_k(r)\frac{r}{n}\log_2\frac{n}{r} \approx \log_2(n) - \sum_{r=1}^{n} c_k(r)\;r\;\log_2(r). \tag{2}$$

The second approximation is due to the fact that $\sum_{r=1}^{n} n\,c_k(r)$ must be approximately n.

In (2) we ignore that for a cycle node we have to count not only the incoming tree nodes but also the incoming cycle node. However, for large n the number of cycle points, $\sqrt{\pi n/2}$, is only a small fraction of all points. Therefore the introduced error is negligible if k, and thus the number of points which reach a cycle is not too large. We see that, as in the case of image points, the estimated entropy loss is almost independent of n.

To obtain $c_k(r)$ we build the generating function of the functional graph by marking all nodes in the tree which are reached by r nodes after k iterations. Such a node must have j children, $1 \le j \le r$, where each of them can be reached by $i_1, \ldots, i_j$ nodes, respectively, after $k-1$ iteration. It must hold that $i_1 + \ldots + i_j = r$. In addition, the tree can have an arbitrary number of children which are trees of length at most $k-1$. By applying the singularity analysis as in [FO90] we get the expected value for $c_k(r)$

$$c_k(r) = \frac{1}{e} f_1(r) \sum_{[i_1,\ldots,i_j]\in Par(r)} c_{k-1}(i_1)\ldots c_{k-1}(i_j) f_2([i_1,\ldots,i_j]) \tag{3}$$

where $c_1(r) = 1/r!e$ is the number of r-nodes $rn(r)$, $f_1(r) = e^{f_1(r-1)/e}$ with $f_1(1) = 1$, $Par(r)$ is the set of all partitions of the integer r[1] and $f_2([i_1,\ldots,i_j])$ is a correction term[2]. This approach can be extended to estimate the entropy not only in the uniform case of the initial distribution but also for any arbitrary distribution or the special case where exactly $m < n$ points have probability $1/m$.

In practice it is not easy to calculate (3) exactly. However, for small k the value $c_k(r)$ decreases fast with r and it is sufficient, for an approximation, to calculate the sum only for $r \le maxR$. In Table 1 we compare the entropy loss approximated with our method for different values of $maxR$, with the lower bound of the loss given by the number of image points and the empirical results from the experiment presented in Figure 1. We can see that for small number of k we reach a much better approximation than the upper bound. However, for bigger values of k this method gets computationally too expensive.

3 Collisions Attacks

In [HK05], Hong and Kim state that the update function of the first version of the MICKEY stream cipher [BD05] behaves almost like a random function with regard to the entropy loss

[1] For example for $r = 4$ we get $\{[1,1,1,1],[1,1,2],[2,2],[1,3],[4]\}$.

[2] If there are some $i_{x_1},\ldots,i_{x_\ell}$ with $1 \le x_1 < \ldots < x_\ell \le j$ and $i_{x_1} = \ldots = i_{x_\ell}$ we have to multiply by a factor $1/\ell!$ to compensate this repeated appearance, e.g. $f_2([1,1,1,1,2,2,3]) = \frac{1}{4!2!1!}$.

k	0	1	2	3	4	5	6	7	8
empirical, $n = 2^{16}$	0.0000	0.8272	1.3456	1.7252	2.0251	2.2731	2.4846	2.6690	2.8324
image points	0.0000	0.6617	1.0938	1.4186	1.6800	1.8993	2.0883	2.2546	2.4032
$maxR = 30$	0.0000	0.8272	1.3457	1.7254	2.0253	2.2727	2.4810	2.6561	2.8004
$maxR = 50$	0.0000	0.8272	1.3457	1.7254	2.0253	2.2734	2.4849	2.6693	2.8324

Table 1: Comparison of different methods to estimate the entropy loss.

and the number of image points. They present two collision attacks, however without detailed complexity analysis, which try to exploit the entropy loss of the state. These attacks are directly applicable on our model.

Their idea was that a reduced entropy leads to an increased probability of a collision. Once we have found a collision, i.e. two states are equal although they have started with different initial values, we know that the subsequent output streams are identical. Due to the birthday paradox, we assume that with an entropy of m-bits we reach a collision, with high probability, by choosing $2^{\frac{m}{2}}$ different states.

In this section we present the two attacks introduced in [HK05]. We give a detailed complexity analysis and compare the results to the attempt of finding a collision directly within the initial states. Such a search has a time, space and data complexity of $O(\sqrt{n})$.

3.1 States After k Iterations

The first attack of Hong and Kim chooses randomly m different initial states, applies k times φ on each of them and searches a collision in the m resulting states. The authors conjecture, based on experimental results, that the upper bound of the entropy after k iterations is about $\log_2(n) - \log_2(k) + 1$. Thus, with high probability we find a collision if $m > 2^{(\log_2(n)-\log_2(k)+1)/2} = \sqrt{2n/k}$.

This attack stores only the last value of the iterations and searches for a collision within this set. This leads to a time and space complexity of $m = O(\sqrt{n/k})$. However, we have to apply k times φ, on each of the chosen initial states, which results in a data complexity of $mk = O(\sqrt{kn})$. This means that we increase the data complexity by the same factor as we decrease the time and space complexity.

The question remains, if there exist any circumstances under which we can use this approach without increasing the data complexity. The answer is yes if the stream cipher uses a set of update functions which loose more than $2\log_2(k)$ bits of entropy after k iterations. Such a characteristic would imply that we do not really use random functions to update our state, however, the principle of the attack stays the same.

3.2 Including Intermediate States

The second attack in [HK05] is equivalent to applying $2k-1$ times φ on m different starting values and searching for a collision in all intermediate states from the k-th up to the $(2k-1)$-th iteration. Since after $k-1$ iterations we have about $\log_2(n) - \log_2(k) + 1$ bits of entropy, Hong and Kim assume that we need a m such that $mk \approx \sqrt{n/k}$. They state that this result would be a bit too optimistic since collisions within a row normally do not appear in a practical stream cipher. However, they claim that this approach still represents a realistic threat for the MICKEY stream cipher. We will show that, contrary to their conjecture, this attack has about the same complexities as a direct collision search in the starting points.

Let us take all the $2km$ intermediate states for the $2k-1$ iterations. If there is no collision in this set then there is also no collision in the km states considered by the attack. Thus, the probability of an successful attack is even smaller than $1 - Pr[noColTotal]$, where $Pr[noColTotal]$

is the probability of no collision in the set of $2km$ states. By means of only counting arguments, without using any entropy estimation, we can show that

$$Pr[noColTotal] = \frac{n(n-1)\dots(n-2km+1)}{n^{2km}} \tag{4}$$

where the probability is taken over all functions $\varphi \in \mathcal{F}_n$ and all possible starting values. This is exactly the probability of no collision in $2km$ random points which means that we need at least an m such that $2km = O(\sqrt{n})$. This leads to a time, space and data complexity in the range of $O(\sqrt{n})$. A summary of the complexities of all attacks can be found in the following table. We see that either the time complexity is still $O(\sqrt{n})$ or it is reduced by the same factor as the data complexity is increased.

Remark. By applying the known technique of distinguished points we can reduce the space complexity in the second attack to $O(c\sqrt{n})$ with $0 < c < 1$, yet, the time complexity stays the same. However, one has to consider that the proportion of distinguished points c must not be too small to guarantee that, with high probability, in each row we reach a distinguished point.

attack	time complexity	space complexity	data complexity
after k iterations	$O(\sqrt{n/k})$	$O(\sqrt{n/k})$	$O(\sqrt{kn})$
with interm. states	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
with interm. states and DPs	$O(\sqrt{n})$	$O(c\sqrt{n})$	$O(\sqrt{n})$

4 Conclusion

We have introduced a new method to estimate the state entropy of a stream cipher which uses a random update function. This estimator is based on the number of points that produce the same value after k iterations. Its computation is too expensive for large numbers of iterations, however, for small numbers it is much more precise than the upper bound given by the number of image points. The same idea can be applied to study the entropy of a Feedback with Carry Shift Register (FCSR) [GK02]. If we have an n bits main register and c carry bits, then the initial entropy is $c + n$. By a similar approach we can compute the final entropy and can show that the entropy loss after one iteration is already $\frac{c}{2}$ bits.

We have also examined the two collision attacks proposed in [HK05]. We pointed out that the first attack improves the time and memory complexity at the cost of significantly increasing the data complexity. We proved that the complexity of the second attack is equivalent to the complexity of a direct collision search in the starting values. The use of random functions in a stream cipher introduces the problem of entropy loss. However, the proposed attacks based on this weakness are less effective than expected.

References

[GK02] M. Goresky and A. Klapper. Fibonacci and Galois representations of feedback-with-carry shift registers. *IEEE Transactions on Information Theory*, pages 2826–2836, 2002.

[BD05] S. Babbage and M. Dodd. The stream cipher MICKEY (version 1). eSTREAM, ECRYPT Stream Cipher Project, Report 2005/015, 2005. `http://www.ecrypt.eu.org/stream`.

[FO90] P. Flajolet and A. M. Odlyzko. Random mapping statistics. *Advances in Cryptology, Proc. Eurocrypt'98*, pages 329–354, 1990.

[HK05] J. Hong and W.H. Kim. TMD-tradeoff and state entropy loss considerations of stream-cipher MICKEY. *INDOCRYPT*, pages 169–182, 2005.

[SS02] N. Sendrier and A. Seznec. HArdware Volatile Entropy Gathering and Expansion: generating unpredictable random numbers at user level. Technical report, INRIA, 2002.

Cryptanalysis of Achterbahn-128/80 with a new keystream limitation

Mara Naya-Plasencia

Projet CODES,INRIA Rocquencourt, France
http://www-rocq.inria.fr/codes/
Maria.Naya_Plasencia@inria.fr

Abstract

This paper presents two key-recovery attacks against the modification to Achterbahn-128/80 proposed by the authors at SASC 2007 due to the previous attacks. The 80-bit variant, Achterbahn-80, was limited to produce at most 2^{52} bits of keystream with the same pair of key and IV, while Achterbahn-128 was limited to 2^{56}. The attack against Achterbahn-80 has complexity 2^{67} and needs fewer than 2^{52} bits of keystream, and the one against Achterbahn-128 has complexity 2^{104} and needs fewer than 2^{56} keystream bits. These attacks are based on the previous ones. The attack against Achterbahn-80 uses a new idea which provides a trade-off between the keystream length and the computational complexity when decimating.

Keywords. eSTREAM, stream cipher, Achterbahn, cryptanalysis, correlation attack, linear approximation, parity check, key-recovery attack.

1 Introduction

Achterbahn [GGK05, GGK06b] is a stream cipher proposal submitted to the eSTREAM project. After the cryptanalysis of the first two versions [JMM06, HJ06b], it has moved on to a new one called Achterbahn-128/80 [GGK06a] published in June 2006. Achterbahn-128/80 corresponds to two keystream generators with key sizes of 128 bits and 80 bits. They consistis respectively of 13 and 11 primitive non linear feed back shift registers of length $L_i = 21 + i$ for register i, and a boolean combining function(F for 128 and G for 80) that takes as inputs the output of the NLFSR's and generates the keystream. Their maximal keystream length was limited to 2^{63}.

As this version was attacked [HJ06a, NP06], the authors proposed a new limitation of the keystream length [GGK07], which was 2^{52} for Achterbahn-80 and 2^{56} for Achterbahn-128. We present here two attacks against both generators, which are based on the previous ones. The attack against the 80-bit variant, Achterbahn-80, has complexity 2^{67} and needs fewer than 2^{52} keystream bits. The attack against Achterbahn-128 requires 2^{104} operations and fewer than 2^{56} keystream bits.

2 Distinguishing attack against Achterbahn-80

Now, we describe a new attack against Achterbahn-80 with a complexity of 2^{67} where a linear approximation of the output function is considered. The attack is a distinguishing attack but it also allows to recover the initial states of certain constituent registers.

This attack is very similar to the previous attack against Achterbahn-80. Our attack exploits the following linear approximation of the combining function G:

$$\ell(x_1, \ldots, x_{11}) = x_1 + x_3 + x_4 + x_5 + x_6 + x_7 + x_{10}.$$

Since G is 6-resilient, ℓ is the best approximation by a 7-variable function. For $\ell(t) = x_1(t) + x_3(t) + x_4(t) + x_5(t) + x_6(t) + x_7(t) + x_{10}(t)$, the keystream $(S(t))_{t\geq 0}$ satisfies $\Pr[S(t) = \ell(t)] = \frac{1}{2}(1 - 2^{-3})$.

Parity-checks. Let us build a parity-check as follows:

$$\ell\ell(t) = \ell(t) + \ell(t + T_3T_7) + \ell(t + T_4T_5) + \ell(t + T_3T_7 + T_4T_5).$$

The terms containing the sequences x_3, x_4, x_5, x_7 vanish in $\ell\ell(t)$, so $\ell\ell(t)$ depends exclusively on the sequences x_1, x_6 and x_{10}. The period T_4T_5 is 2^{51} and the period T_3T_7 is smaller than 2^{49} as T_3 and T_7 have common factors, so to build those parity checks we need less than the maximal keystream length allowed.

Adding four times the approximation has the effect of multiplying the bias four times, so the bias of $\sigma(t) = S(t) + S(t + T_7T_3) + S(t + T_4T_5) + S(t + T_7T_3 + T_4T_5)$ where $(S(t))_{t\geq 0}$ is the keystream, is $2^{-4\times 3}$. We now decimate $\sigma(t)$ by the period of the register 1, which is involved in the parity-check, so we create like this a new parity-check:

$$\sigma'(t) = \sigma(t(2^{22} - 1)).$$

Then, if we did as in the previous attack and we performed an exhaustive search for the initial states of registers 6 and 10, we would need $2^{3\times 4\times 2} \times 2 \times (58 - 2) \times \ln(2) = 2^{30.29}$ parity-checks $\sigma'(t)$ to detect this bias. As we are decimating by the period of the register 1, we would need $2^{30.29} \times 2^{22} = 2^{52.29}$ keystream bits to perform the attack, and it is over the limitation, so we cannot do that.

What we are going to do, instead of taking only the first bit of the keystream and of decimating by the period of the first register $2^{30.29}$ times, consists in considering the first four consecutive shifts of the keystream and for each one, we are going to obtain a sequence of $\frac{2^{30.29}}{4} = 2^{28.29}$ bits by decimating it by the period of the first register $2^{28.29}$ times. Thus, we consider the first $2^{50.29}$ bits of the keystream and we compute the $4 \times 2^{28.29} = 2^{30.29}$ parity checks:
$S(t(2^{22} - 1) + i) + S(t(2^{22} - 1) + i + T_7T_3) + S(t(2^{22} - 1) + i + T_4T_5)+$
$S(t(2^{22} - 1) + i + T_7T_3 + T_4T_5)$
for $i \in \{0, \ldots, 3\}$ and $0 \leq t < 2^{28.29}$. This way, the number of keystream bits that we need is reduced to $2^{28.29} \times 2^{22} = 2^{50.29}$ and respects the maximal keystream length permitted. This is going to change time complexity as now we will have to perform an exhaustive search on the four following bits generated by register 1:

$$z_i = x_1(i) + x_1(i + T_7T_3) + x_1(i + T_4T_5) + x_1(i + T_7T_3 + T_4T_5)$$

for $i \in \{0, \ldots, 3\}$. As we do not care about the real value of those bits but about the differences $(z_i + z_0) \forall i \in \{1, 2, 3\}$, we are going to evaluate the 2^3 possible states only. Thus, we perform an exhaustive search over the registers 6 and 10, adapting to our new situation the algorithm introduced in [NP06]. We will have to compute, for each one of the previously mentioned sequences, so for each $i \in \{0, 1, 2, 3\}$, the following sum:

$$\begin{aligned}
&\sum_{t'=0}^{2^{28.29}-1} \sigma(t'T_1 + i) \oplus \ell\ell(t'T_1 + i) \\
= &\sum_{k=0}^{T'} \sum_{t=0}^{2} \sigma((T_6t + k)T_1 + i) \oplus \ell\ell((T_6t + k)T_1 + i) \\
+ &\sum_{k=T'+1}^{T_6-1} \sum_{t=0}^{1} \sigma((T_6t + k)T_1 + i) \oplus \ell\ell((T_6t + k)T_1 + i) \\
= &\sum_{k=0}^{T'} \sum_{t=0}^{2} \sigma((T_6t + k)T_1 + i) \oplus \sigma_{10}((T_6t + k)T_1 + i) \oplus \sigma_6((T_6t + k)T_1 + i) \\
+ &\sum_{k=T'+1}^{T_6-1} \sum_{t=0}^{1} \sigma((T_6t + k)T_1 + i) \oplus \sigma_{10}((T_6t + k)T_1 + i) \oplus \sigma_6((T_6t + k)T_1 + i)
\end{aligned}$$

$$\begin{aligned}
= &\sum_{k=0}^{T'} \left[(\sigma_6(kT_1 + i) \oplus 1) \left(\sum_{t=0}^{2} \sigma((T_6t + k)T_1 + i) \oplus \sigma_{10}((T_6t + k)T_1 + i) \right) \right. \\
+ &\left. \sigma_6(kT_1 + i) \left(3 - \sum_{t=0}^{2} \sigma((T_6t + k)T_1 + i) \oplus \sigma_{10}((T_6t + k)T_1 + i) \right) \right] \\
+ &\sum_{k=T'}^{T_6-1} \left[(\sigma_6(kT_1 + i) \oplus 1) \left(\sum_{t=0}^{1} \sigma((T_6t + k)T_1 + i) \oplus \sigma_{10}((T_6t + k)T_1 + i) \right) \right. \\
+ &\left. \sigma_6(kT_1 + i) \left(2 - \sum_{t=0}^{1} \sigma((T_6t + k)T_1 + i) \oplus \sigma_{10}((T_6t + k)T_1 + i) \right) \right],
\end{aligned}$$

where $\sigma(t), \sigma_6(t)$ and $\sigma_{10}(t)$ are the parity checks computed at the instant t with the keystream, the sequence generated by the register 6 and the one generated by the register 10 respectively, and where

$$T' = 2^{28.29} - 2 \times T_6 = 2^{28.29} - 2^{28} + 2 = 2^{25.83}.$$

The sum can be written the above way since $\sigma_6((T_6t + k)T_1 + i)$ is constant for fixed values of k and i. We are going to see what we have got to do at this point:

- We choose an initial state for register 6, e.g. the all-one initial state. We compute and save a binary vector V_6 of length T_6, $V_6[k] = \sigma_6(k)$, where the sequence with whom we are computing $\sigma_6(k)$ is generated from the chosen initial state. The complexity of this state is $T_6 \times 2^2$ operations.
- For each possible initial state of register 10 (so 2^{31-1} possibilities):
  - we compute and save four vectors $V_{10,i}$, where $i \in \{0, 1, 2, 3\}$, each one composed of T_6 integers of 2 bits.

    $$V_{10,i}[k] = \sum_{t=0}^{m} \sigma((T_6t + k)T_1 + i) \oplus \sigma_{10}((T_6t + k)T_1 + i),$$

    where $m = 2$ if $k \leq T'$ and $m = 1$ if $k > T'$. The complexity of this state is:

    $$2^2 \left(3 \times 2^{25.83} + 2(2^{27} - 1 - 2^{25.83})\right)(2^3 + 2) \quad = \quad 2^2 \times 2^{28.29} 2^{3.3} = 2^{33.69}$$

    for each possible initial state of register 10, where 2^2 is the number of vectors that we are computing, 2^3 corresponds to the number of operations required for computing each $(\sigma(t) + \sigma_{10}(t))$ and $2^{28.29} \times 2$ is the cost of summing up $2^{28.29}$ integers of 2 bits.
  - For each possible p from 0 to $T_6 - 1$:
    - we define $V_{6,i}$ of length T_6, $\forall i \in \{0, 1, 2, 3\}$: $V_{6,i}[k] = V_6[k + p + i \mod T_6]$. Actually, $(V_{6,i}[k])_{k<T_6}$ corresponds to $(\sigma_6(k))_{k<T_6}$ when the initial state of register 6 corresponds to the internal state obtained after clocking $(i + p)$ times register 6 from the all-one initial state.
    - With the eight vectors that we have obtained $(V_{10,0}, \ldots, V_{10,3}, V_{6,0}, \ldots, V_{6,3})$, we compute for each $i \in \{0, 1, 2, 3\}$:

      $$\begin{aligned}
      W_i \quad = \quad &\sum_{k=0}^{T'} \left[(V_{6,i}[k] \oplus 1)\, V_{10,i}[k] + V_{6,i}[k]\, (3 - V_{10,i}[k]) \right] + \\
      &\sum_{k=T'+1}^{T_6-1} \left[(V_{6,i}[k] \oplus 1)\, V_{10,i}[k] + V_{6,i}[k]\, (2 - V_{10,i}[k]) \right].
      \end{aligned}$$

      Now, we compute the sum of the four W_i, so for all the 2^3 possible states of the bits $z_0, \ldots, z_3$ coming from register 1, and considering z_0 as positive:

$$W_0 \pm W_1 \pm W_2 \pm W_3.$$

When we do this with the correct initial states of registers 6 and 10, we will find the expected bias at one of the 2^3 possible sums.

The complexity of this point would be, for each p $2^2 \times T_6 \times 8 \times 2^3 = 2^{35}$, so $2^{35} \times 2^{27} = 2^{62}$. But we can speed up the process: we define two new vectors,

$$V_6[k'] = V_{6,j}[k],\ V_{10}[k'] = V_{10,j}[k] + ct$$

where $k' = jT_6 - 1 + k$, $k' \in \{0, \ldots, 4T_6 - 1\}$, and $ct = 0$ if $k \leq T'$ and $ct = 0.5$ if $k > T'$.

With those two vectors we are going to compute:

$$\sum_{k'=0}^{4T_6-1} (-1)^{V_6[k'+p]} \left(V_{10}[k] - \frac{3}{2}\right) + 4 \times (T' \times 1.5 + (T_6 - T') \times 1).$$

The issue is now to find the p that maximizes this sum, this is the same as computing the maximum of the crosscorrelation of two sequences of length $4T_6$. We can do that efficiently using a fast Fourier transform as explained in [Bla85, pages 306-312]. The final complexity for computing this sum will be in $4T_6 \log_2(4T_6)$. We have not finished yet, as we have to do this for each one of the possible relative states of the z_i, and we can do that by recomputing $V_6[k']$, which will have a low complexity as all we have to do is to xor the bits of the states corresponding to the $i = j$ when $z_i \neq z_0$. Thus, the total complexity of this state will be $2^3 \times 4T_6 \log_2(4T_6) \approx 2^{37}$

The time complexity is going to be, finally $2^{L_{10}-1} \times \left[2^{33.69} + 4T_6 \log_2 4T_6 \times 2^3\right] + T_6 \times 2^2 = 2^{67}$. The number of keystream bits that we need is $2^{28.29} \times T_1 + T_3T_7 + T_4T_5 = 2^{50.29} + 2^{48.1} + 2^{51} < 2^{52}$.

3 Distinguishing attack against Achterbahn-128

Now, we present a distinguishing attack against the 128-bit version of Achterbahn which also recovers the initial states of two registers.

We consider the following approximation of the combining function F:

$$\ell(x_0, \ldots, x_{12}) = x_0 + x_1 + x_2 + x_3 + x_4 + x_7 + x_8 + x_9 + x_{10}.$$

Then, for $\ell(t) = x_0(t) + x_1(t) + x_2(t) + x_3(t) + x_4(t) + x_7(t) + x_8(t) + x_9(t) + x_{10}(t)$, we have $\Pr[S(t) = \ell(t)] = \frac{1}{2}(1 + 2^{-3})$.

Parity-checks. If we build a parity check as follows:

$$\ell\ell\ell(t) = \sum_{\tau \in \langle T_{3,8}, T_{1,10}, T_{2,9} \rangle} \ell(t + \tau),$$

the terms containing the sequences x_1, x_2, x_3, x_8, x_9, x_{10} will disappear from $\ell\ell\ell(t)$, so $\ell\ell\ell(t)$ depends exclusively on the sequences x_0, x_4 and x_7:

$$\ell\ell\ell(t) = \sum_{\tau \in \langle T_{3,8}, T_{1,10}, T_{2,9} \rangle} x_0(t + \tau) + x_4(t + \tau) + x_7(t + \tau) = \sigma_0(t) + \sigma_4(t) + \sigma_7(t),$$

where $\sigma_0(t)$, $\sigma_4(t)$ and $\sigma_7(t)$ are the parity-checks calculated on the sequences generated by NLFSRs 0, 4 and 7. Adding eight times the approximation has the effect of multiplying the bias eight times, so the bias of $\sigma(t) = \sum_{\tau \in \langle T_{3,8}, T_{1,10}, T_{2,9} \rangle} S(t + \tau)$, where $(S(t))_{t \geq 0}$ is the keystream, is $2^{-8 \times 3}$. So:

$$\Pr[\sigma(t) + \sigma_0(t) + \sigma_4(t) + \sigma_7(t) = 1] = \frac{1}{2}(1 - \varepsilon^8).$$

This means that we need $2^{3 \times 8 \times 2} \times 2 \times (74 - 3) \times \ln(2) = 2^{54.63}$ values of $\sigma(t) + \sigma_0(t) + \sigma_4(t) + \sigma_7(t)$ to detect this bias, when we perform an exhaustive search on registers 0, 4 and 7.

We are going to use the algorithm proposed previously for the attack of Achterbahn-128 for computing the sum $\sigma(t) + \sigma_0(t) + \sigma_4(t) + \sigma_7(t)$ over all values of t. This algorithm has a lower complexity than an exhaustive search for the initial states of the registers 0, 4 and 7 simultaneously. We are going to use it considering register 0 and register 4 together.

The complexity is going to be, finally $2^{L_0-1} \times 2^{L_4-1} \times \left[2^{54.63} \times \left(2^4 + 2^{4.7}\right) + O(T_7 \log T_7)\right] + T_7 \times 2^3 = 2^{104}$.

The length of keystream needed is: $2^{54.63} + T_{1,10} + T_{2,9} + T_{3,8} = 2^{54.63} + 2^{53} + 2^{53} + 2^{53} < 2^{56}$ bits.

4 Recovering the key

As explained in the previous attacks, with a variant of a meet-in-the-middle attack we can recover the key once we have found the initial state of some registers, and the complexity is going to be smaller than the one needed to perform the previously described distinguishing attack that we need to get the initial state of several registers. So the complexity of the total key-recovery attack is going to be the same one as for the distinguishing attacks.

5 Conclusion

We have proposed an attack against Achterbahn-80 in 2^{67} where fewer than 2^{52} bits are needed. An attack against Achterbahn-128 is also proposed in 2^{104} where fewer than 2^{56} bits of keystream are required. After that we can recover the key of Achterbahn-80 with a complexity of 2^{40} in time and 2^{41} in memory (the time complexity is less than for the distinguishing part of the attack). For Achterbahn-128 we can recover the key with a complexity of 2^{73} in time and 2^{48} in memory.

References

[Bla85] R. E. Blahut. *Fast Algorithms for Digital Signal Processing*. Addison Wesley, 1985.

[GGK05] B. M. Gammel, R. Gottfert, and O. Kniffler. The Achterbahn stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/002, 2005. `http://www.ecrypt.eu.org/stream/ciphers/achterbahn/achterbahn.pdf`.

[GGK06a] B. M. Gammel, R. Gottfert, and O. Kniffler. Achterbahn-128/80. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/001, 2006. `http://www.ecrypt.eu.org/stream/p2ciphers/achterbahn/achterbahn_p2.pdf`.

[GGK06b] B. M. Gammel, R. Gottfert, and O. Kniffler. Status of Achterbahn and tweaks. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/027, 2006. `http://www.ecrypt.eu.org/stream/papersdir/2006/027.pdf`.

[GGK07] B. M. Gammel, R. Gottfert, and O. Kniffler. Achterbahn-128/80: Design and analysis. In *ECRYPT Network of Excellence - SASC Workshop Record*, pages 152–165, 2007.

[HJ06a] M. Hell and T. Johansson. Cryptanalysis of Achterbahn-128/80. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/054, 2006. `http://www.ecrypt.eu.org/stream/papersdir/2006/054.pdf`.

[HJ06b] M. Hell and T. Johansson. Cryptanalysis of Achterbahn-version 2. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/042, 2006. `http://www.ecrypt.eu.org/stream/ciphers/achterbahn/achterbahn.pdf`.

[JMM06] T. Johansson, W. Meier, and F. Muller. Cryptanalysis of Achterbahn. In *Advances in Cryptology - FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2007.

[NP06] M. Naya-Plasencia. Cryptanalysis of Achterbahn-128/80. In *Advances in Cryptology - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, *to appear*. Springer, 2006.

Using Two-Steps Hash Function to Support Trustworthy Signing in XML Signature

Sebastian Gajek*, Lijun Liao† and Jörg Schwenk‡

Horst-Görtz Institute of IT-Security, Ruhr-Univeristy Bochum
*sebastian.gajek@nds.rub.de
†lijun.liao@nds.rub.de
‡joerg.schwenk@nds.rub.de

Abstract

Digital signatures created on smart card systems are employed in many current e-commerce applications. But even when the underlying cryptographic methods are sufficiently strong and the smart card system are sufficient secure, attacks by Trojan horse programs on digital signatures are still possible. In this paper, we propose an approach to create XML signatures trustworthily in smart card system which has at least the reader with display and PIN pad in untrustworthy environments. We have implemented a prototype of the approach.

Keywords. XML Signature, Smart Card, Hash function, E-Commerce

1 Proposed Solution

In our approach we define a new canonicalization method (in Sect. 1.2) which canonicalizes the `<SignedInfo>` in XML Signature [1] first according to some standard canonicalization method and then appends the sensitive information retrieved from the content that is signed. When creating the signature, we compute then the hash value using our new two-steps hash function (in Sect. 1.1).

1.1 Two-Steps Hash Function

Most hash functions use Merkle-Damgård form as shown in Fig. 1. A hash function has a fixed initialization vector IV. The input message m is split to n blocks, the last block is appended with zeros followed with the message length l (a new block for l is added if the last block has not enough place for l). Finally, each block has α octets, namely of block length α. Each block is then compressed with a compression function f, which needs an initialization vector, for the first f is IV, for other f is the result of previous f. A compression function that is initiated with the initialization vector iv and compresses the data x is denoted as $f(iv, x)$. The finalization function $final$ takes the result of the last compression function as input and its output is the hash value.

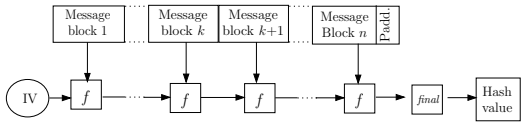


Figure 1: Merkle-Damgård form for hash functions

Let $\Gamma_\Phi = (iv, bl, ll, hash)$ be the notation for the hash function Φ. iv and bl return the initialization vector and block length of Φ, denoted as $iv(\Phi)$ and $bl(\Phi)$, respectively. ll returns the needed octets to represent $bl(\Phi)$, denoted as $ll(\Phi)$. $hash$ computes the hash value according to Φ. Assume that the message is represented by m, $hash(m)$ returns the hash value over m according to Φ. Let $length(x)$ be the number of octets contained in x, $\psi(u, v)$ be u octets with value v, and $||$ is the concatenation of octets. Two padding functions are defined in Definition 1.1, and the hash function in Definition 1.2.

Definition 1.1 (Padding Function) *A padding function that appends α octets zeros and γ octets with value l to the message m is defined as follows:*

$$pad(m, \alpha, l, \gamma) = m \;||\; \psi(\alpha, 0) \;||\; \psi(\gamma, l) \tag{1}$$

Based on this padding function we define the padding function for Γ_Φ as follows:

$$pad_\Phi(m) = pad(m, bl(\Phi) - ((\beta + ll(\Phi)) \bmod bl(\Phi), \beta, ll(\Phi)), \text{ where } \beta = length(m)\ . \tag{2}$$

Definition 1.2 (Hash Function) *Given a hash function system $\Gamma_\Phi = (iv, bl, ll, h)$ and a message m. $pad_\Phi(m) = m_1 \;||\; \cdots \;||\; m_n$, where $length(m_i) = length(m_j)$ for $1 \le i \le j \le n$. Then*

$$hash(m) = final(f_n), \text{ where } f_0 = iv(\Phi), f_i = f(f_{i-1}, m_i) \text{ for } 1 \le i \le n\ . \tag{3}$$

Since at the last block, the message length with leading zeros are appended, we cannot divide a message m to m_I and m_{II}, i.e. $m = m_I || m_{II}$, and compute the hash value of m as follows $hash(m) = g(hash(m_I), m_{II})$, where g is any function. However, with slightly modification we can compute hash value of a message in two steps as above. We call this modified version as two-steps hash function. It contains two steps: the first step $hash_I$ and the second step $hash_{II}$.

In two-steps hash functions, the $length(m_I)$ must be multiple of the block length of Φ. We compute first $hash_I$ as defined in Definition 1.3; we compute then $hash_{II}$, which takes the result and the processed blocks of $hash_I$ and m_{II} as input, as defined in Definition 1.4.

Definition 1.3 ($hash_I$ of two-steps hash function) *Given a hash function system $\Gamma_\Phi = (iv, bl, ll, hash)$ and a message m_I with $k \cdot bl(\Phi)$ octets. $m_I = m_{I_1} \;||\; \cdots \;||\; m_{I_k}$, where $length(m_{I_i}) = length(m_{I_j})$ for $1 \le i \le j \le k$. Then*

$$hash_I(m_I) = f_{I_i}, \text{ where } f_{I_0} = iv(\Phi), f_{I_i} = f(f_{I_{i-1}}, m_{I_i}) \text{ for } 1 \le i \le k\ . \tag{4}$$

In $hash_{II}$, m_{II} is first padded according to $pad(m_{II}, n, k \cdot bl(\Phi) + length(m_{II}), ll(\Phi))$, where $n = bl(\Phi) - ((length(m_{II}) + ll(\Phi)) \bmod bl(\Phi))$. For simplify we denote it as $pad_{\Phi_{II}}(m_{II}, k)$.

Definition 1.4 ($hash_{II}$ of two-steps hash function) *Given a hash function system $\Gamma_\Phi = (iv, bl, ll, hash)$, k (the number of processed blocks in h_I), ρ (the result of h_I), and a message m_{II}. $pad_{\Phi_{II}}(m_{II}, k) = m_{II_1} \;||\; \cdots \;||\; m_{II_q}$, where $length(m_{II_i}) = length(m_{II_j})$ for $1 \le i \le j \le q$. Then*

$$hash_{II}(m_{II}, \rho, k) = final(f_{II_q}), \text{ where } f_{II_0} = \rho, f_{II_i} = f(f_{II_{i-1}}, m_{II_i}) \text{ for } 1 \le i \le q\ . \tag{5}$$

Based on the definitions above, we get Theorem 1.5.

Theorem 1.5 *Given a hash function system $\Gamma_\Phi = (iv, bl, ll, hash)$ and a message m. Let m_I be the first $k \cdot bl(\Phi)$ octets of a message m, and m_{II} be the octets of m from the position $k \cdot bl(\Phi) + 1$, then*

$$hash(m) = hash_{II}(m_{II}, hash_I(m_I), k)\ . \tag{6}$$

Proof. *Clearly $pad_\Phi(m) = m_I \;||\; pad_{\Phi_{II}}(m_I, i)$, which gives*

$$m_{I_i} = m_i, \text{ where } 1 \le i \le k \tag{7}$$

and

$$m_{II_i} = m_{k+i}, where\ 1 \leq i \leq q\ . \quad (8)$$

Equation 7 gives: $hash_I(m_I) = f(f_{I_{k-1}}, m_{I_k}) = f(f_{k-1}, m_k) = f_k$*, which gives together with (8):* $f_{II_1} = f(f_{II_0}, m_{II_1}) = f(f_k, m_{k+1}) = f_{k+1}$*, and so on till* $f_{II_q} = f_{k+q}$*. Finally we get*

$$hash(m) = final(f_{k+q}) = final(f_{II_q}) = hash_{II}(m_{II}, hash_I(m_I), k)\ .$$

□

1.2 New Canonicalization Method in XML Signature

XML Signature is specified by the `<Signature>`, as in lines 7–15 in Fig. 2. For details about XML Signature please refer to the specification [1]. The `<SignedInfo>` is first canonicalized and the hash value is then computed over the canonicalization result. Considering the example in Fig. 2, the `<Remittance>` with the attribute `@Id="abc"` (in lines 2–5) is protected by the `<Reference>` (in lines 11–13) via the attribute `URI="#abc"`. The `<SignedInfo>` is then canonicalized according to the algorithm EXC-C14N [2], specified by the URI `http://www.w3.org/2001/10/xml-exc-c14n#`. EXC-C14N is a serialization of XML, when applied to a subdocument, includes only the related namespace declarations.

```
1 <BankTransfer>
2  <Remittance Id='abc'>
3   <Remittee><Account>123456</Account><Name>Bob</Name>
4    <BankCode>11122200</BankCode><Bankname>Some Bank</Bankname></Remittee>
5   <Amount>428.00 EUR</Amount><Purpose>Some Purpose</Purpose>
6  </Remittance>
7  <Signature>
8   <SignedInfo>
9    <CanonicalizationMethod Algorithm='.../xml-exc-c14n#'/>
10   <SignatureMethod Algorithm="...#rsa-sha1"/>
11   <Reference URI='#abc' Id='Ref-1'>
12    <DigestMethod Algorithm='...#sha1'/><DigestValue>a...A=</DigestValue>
13   </Reference></SignedInfo>
14  <SignatureValue>mUJ...yEg=</SignatureValue><KeyInfo>...</KeyInfo>
15 </Signature>
16 </BankTransfer>
```

Figure 2: A bank transfer with XML signature

Let $c14n(e)$ be the function that canonicalizes the element e according to EXC-C14N, then the hash value is computed over $\phi = c14n($`<SignedInfo>`$)$, and $hash(\phi)$ is finally signed. In the current systems that using XML signature, $hash(\phi)$ is sent from the client to the smart card system and is signed there.

For our approach, we define a new canonicalization method `EXT-CANON`, which is identified by the URI `http://www.example.org/ext-canon`. It has the child elements `<Base>` and `<Appendent>`. `<Base>` specifies the canonicalization method which is first applied to the `<SignedInfo>` element. The `<Appendent>` specifies how the result is appended. The child element `<DigestBlockLength>` specifies the block length of the hash function, e.g. 64 octes for SHA1; the child element `<Transform>` specifies how the resource referenced by the `<Reference>`, which is turn referenced by the attribute `@URI` of `<Appendent>`, is transformed. The transformed result is then appended to the canonicalization result.

If we replace the `<CanonicalizationMethod>` in Fig. 2 with the one depicted in Fig. 3. The result after performing the canonicalization method specified in `<Base>` is $\phi = c14n($`<SignedInfo>`$)$, and the length is $l_1 = length(\phi)$. Since $bl = 64$, $\alpha = 64 - (l_1 \bmod 64)$ octets zeros are appended.


```
1 <CanonicalizationMethod Algorithm='http://www.example.org/ext-canon'>
2  <Base URI='.../xml-exc-c14n#'/>
3  <Appendent URI='#Ref-1'><DigestBlockLength>64</DigestBlockLength>
4   <Transform Algorithm='.../REC-xslt-19991116'>
5    <xsl:stylesheet><xsl:output method='text'/>
6     <xsl:template match='Remittance'>
7      Bank:<xsl:value-of select='Remittee/BankCode'/>
8      Account:<xsl:value-of select='Remittee/Account'/>
9      Amount:<xsl:value-of select='Amount'/>
10    </xsl:template></xsl:stylesheet></Transform>
11 </Appendent>
12 </CanonicalizationMethod>
```

Figure 3: An example of new canonicalization method `EXT-CANON`

The `@URI` attribute with value `#Ref-1` references the `<Reference>` in lines 11–13 (in Fig. 2), which in turn references the `<Remittance>` in lines 2–6 (in Fig. 2). We apply the XSLT transform specified in lines 4–10 (in Fig. 3) to the `<Remittance>` element and get the transformation result λ as follows:

```
Bank:11122200
Account:123456
Amount:428.00 EUR
```

The transformation result is then appended, and we get the canonicalization result $m = \phi \mid\mid \psi(k, 0) \mid\mid \lambda$.

1.3 Model

Let $\mathcal{E} = (U, R, C, S, R \leftrightarrow C, C \leftrightarrow S)$ be the notation for the system which is considered in this paper. U denotes the user, R the smart card system (including the smart card and the card reader, and their communication), C the client that communicates with the server and prepares the signature, and S the server that receives the request from the client and verifies the signature. $R \leftrightarrow C$ and $C \leftrightarrow S$ denote the communication between R and C and between C and S, respectively. We will show how to trustworthily create signature in smart cart system under Assumption 1.6.

Assumption 1.6 *$\mathcal{E} = (U, R, C, S, R \leftrightarrow C, C \leftrightarrow S)$ satisfies the following requirements: 1). All underlying cryptographic functions, e.g. the hash function, the signature function, are secure. 2). R and S are secure.*

Figure 4 depicts the outflow of the creation and verification of the `<BankTransfer>` in Fig. 2 with the new canonicalization method in Fig. 3. $R \leftrightarrow C$ and $R \leftrightarrow C$ are considered as secure in Fig. 4. The server and any other verifiers can compute the hash value and verify the signature as usual. The implementation of the two-steps hash function is not needed for the verification process. The only extension is the new transform method `EXT-CANON` should be implented.

If the client C and the communications $R \leftrightarrow C$, $C \leftrightarrow S$ are not secure. Considering the example in Fig. 4 the following attacks are possible:

1. The attacker modifies the bank transfer, e.g. the recipient account is changed from 123456 to 654321, and sent the modified one to the smart card system. Since the

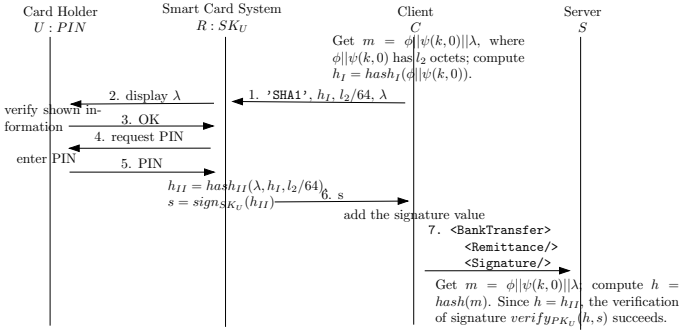


Figure 4: Outflow of signature creation and verification: Only U knows the PIN (R can verify PIN, but does not know PIN), only R knows the secret signing key SK_U with the corresponding public verification key PK_U. The function $verify_{PK_U}(h,s)$ verifies if s is the signature of h which is signed with SK_U.

modified bank transfer is displayed in the card reader, the user can detect the modification and will cancel the signing process.

2. The attacker modifies the bank transfer, but sent the original one to the smart card system. In this case the user will sign the bank transfer. But this modification will be detected by the verifier, since the appending information in the signing process and the verifying process varies.

We have implemented the prototype of our approach in Java [3]. We have simulated the process in Fig. 4 and the attacks described above.

2 Conclusion

In this paper we proposed an efficient approach to create XML signature in smart card based systems trustworthily, even in insecure environments. It can be applied in the case where the sensitive information is of several lines, e.g. online banking and online stock exchange. With our approach, the verification application can verify the signature as usual.

References

[1] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, *XML-Signature Syntax and Processing*, W3C Recommendation, Feb. 2002. [Online]. Available: `http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/`.

[2] J. Boyer, D. Eastlake, and J. Reagle, *Exclusive XML Canonicalization, Version 1.0*, W3C Recommendation, July 2002. [Online]. Available: `http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/`.

[3] L. Liao, "Prototype implementation of trustworthy signing in smart card based system in untrusted enviroments," Apr. 2007. [Online]. Available: `http://www.nds.rub.de/liao/works/scdsign.zip`.

Browser-based Authentication Protocols for Naive Users

Sebastian Gajek[1], Mark Manulis[1], Ahmad-Reza Sadeghi[2] and Jörg Schwenk[1]

Horst Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
[1]{sebastian.gajek,mark.manulis,joerg.schwenk}@nds.rub.de
[2]sadeghi@crypto.rub.de

Abstract

The design of browser-based protocols and in particular browser-based protocols that make use of Web 2.0 technologies bears risks, which are not thoroughly studied. The problem with browser-based protocols is that users are actively involved in the protocol execution: They have to authenticate the server. Recent user studies point out, however, that averaged skilled Internet used are unable to authenticate the server based on current security indicators (e.g., complicated server certificates). In this paper, we propose a browser-based authentication protocol, which relieves these burdens of naive users to authenticate the server. In the extended version of this paper, we plan to prove security of the described protocol within the security model due to Bellare and Rogaway.

Keywords. Protocols, browser, Web 2.0

1 Introduction

The World Wide Web has grown and become a fundamental technology of the information society. To fulfill the growing requirements of the information society, the Web has been continuously improved. In order to highlight an improved form of the World Wide Web, some technology experts recently introduced the term Web 2.0. Technically speaking, Web 2.0 may be expressed in terms of languages and protocols deployed that have become standard extensions of commodity web browsers (e.g., AJAX, SOAP). These technologies allow for implementing, e.g., mashups, blogs, and multimedia streaming applications. They are also appealing for realizing browser-based security protocols (e.g., [12]). In the general case, browser-based protocols are designed for mass-market. This constraints the design of secure browser-based protocols. The past has shown that protocols that (a) build on or extend the TLS protocol, and (b) are password-based proliferated; protocols that deviated from the standard have not been extensively used.

The deployment of browser-based protocols bears risks, which are not thoroughly studied and should become part of rigorous security analysis as done with password-based protocols (e.g., [8, 10, 2]). However, common to the analysis of password-based protocols is that principals are assumed to be machines that follow a strict protocol definition. In the setting of browser-based protocols, however, a human user is an active participant of the protocol. The user is responsible for identifying a honest web site. Consider, for instance, a browser-based authentication protocol running on top of TLS where the user has to enter her password into a web form designed with Flash. Then, the user must intrinsicly ensure that she communicates to the real server. A folklore belief was that the user is a security expert, who is able to authenticate the server. This allowed to idealize the user or refrain from modeling the user and her skills (see, e.g., [8, 10, 2, 7]). However, recent studies [5, 16] point out that averaged-skilled Internet users understand browser's neither server certificates nor security indicators thoroughly. Hence, one have to assume that users are unable to authenticate servers *per se* and become target to man-in-the-middle attacks.

An additional problem is the browser's protocol unawareness [7]. The browser may be partly controlled by browser languages. An adversary may send queries that reveal or alter the internal state of the browser, making browser protocols vulnerable to semantic attacks. For instance, compromise of cookies has led to flaws in Mircosoft's Passport protocol and HTTP redirects allowed to attack SAML's browser artifact profile protocol [11, 6].

The desired properties for browser-based security protocols should include (a) the use of secure technologies, being implemented in commodity web browsers and servers, and (b) take into account skills of security-unaware users. In this paper, we propose a browser-based authentication protocol based on TLS, explicitly taking into account that ordinary Internet users do not understand any complicated security indicators. The key idea is to reverse the roles for authentication. Instead of the user, we enable the server to identify the client based on client certificates. Then, the server ensures that it is connected to a honest client and sends the easy-to-recognize identifier, which user and server securely share (e.g., picture, brand). To authenticate the server, the user has to detect the identifier only; verification of server certificates is trivial. Here we make use of user studies stating that a user more likely recognizes these identifiers than cryptographic parameters or security indicators [17, 4].

2 Related Work

In order to improve server-sided authentication, some work has been done to extend TLS. Steiner et al. [15] propose a password-based extension of TLS. Oppliger et al. [13] propose the notion of TLS session awareness. The authors augment TLS to link users' passwords (or any credentials) to TLS sessions. As a result, servers are able to thwart Man-in-the-Middle attacks, as passwords contain information about the actually involved parties. However, the extensions still assume that the user authenticates the server, otherwise she could enter her password in a password form that was retrieved over an TLS-*unprotected* connection.

Some work has been done to use visual identifiers in protocols instead of cryptographic parameters. Jakobsson et al. [9] propose an oblivious transfer protocol to conjunct password-based mutual authentication with images, i.e., passwords are linked with a sequence of images, which are visual shared secrets. The user has to identify the sequence of images in order to authenticate the server. This protocol is vulnerable to active Man-in-the-Middle attacks. The SiteKey [14] protocol is similar to the presented protocol, however, uses cookies instead of client certificates in order to bind the visual identifiers to user identities. Recently, an attack against SiteKey was published [14].

3 Mutual Authentication Protocol for Naive Users

3.1 Prelimaries

The players that take part in the protocol are the user U, the browser B, and the Server S. U is modeled as a simple Turing machine, meaning the user machine's computational and memorable capabilities are limited. The machine is unable to solve any (cryptographic) hard problems. On the user machine's tape a set of low-entropy passwords is written. A password pw is drawn from a small dictionary $\mathcal{D}$ of size n and shared with the server S. Each password entry is associated with a high-entropy identifier w, which is drawn from the dictionary $\mathcal{W}$. An identifier w shall reflect a string an user may use in order to identify the server. The size of w constraints the computational bounds of a human users. For naive users, we assume that w may be of the size of a memorable passphrase or an image. In order to model the user's abilities to identify w, we introduce an auxiliary predicate function $detect(w, w') : (0,1)^{|w|} \times (0,1)^{|w'|} \rightarrow (\top, \bot)$ that outputs true $\top$ when the identifiers w and w' are indistinguishable, otherwise it outputs false $\bot$. B is modeled as a simplified Turing machine that translates messages sent from server

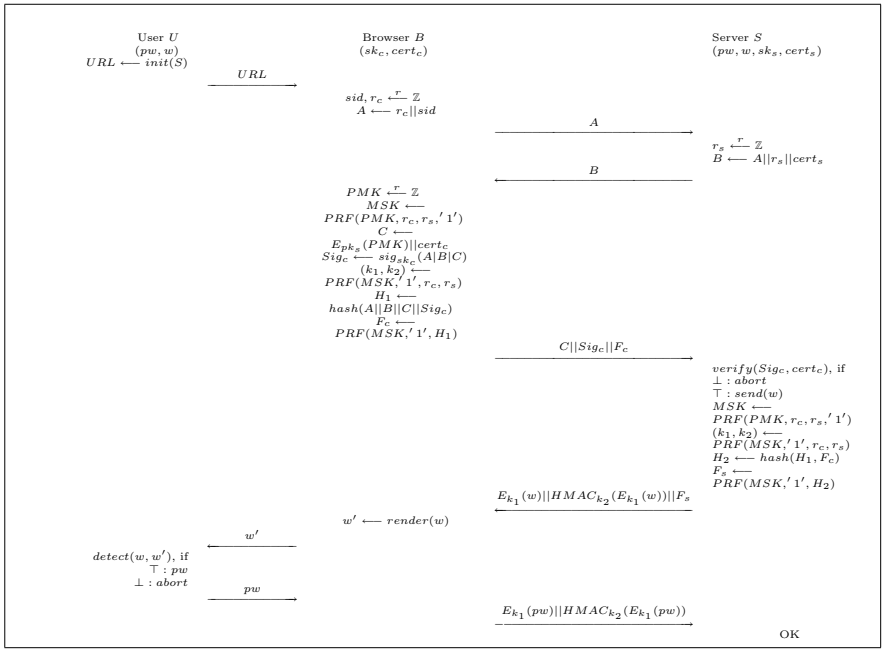


Figure 1: Simplified TLS Key Transport Protocol based on Client Certificates

and user. To capture the browser's ability of processing messages, we use an auxiliary function $render(m) : (0,1)^{|m|} \longrightarrow (0,1)^{|m^*|}$ that takes as input a message m sent from the server of truncated size $|m^*|$.

3.2 Description of the Scheme

The basic building block of the protocol is TLS, more precisely, its handshake protocol. The handshake protocol constitutes many cipher suites. We make use of the cipher suit for mutually authenticated and encrypted key exchange based on client certificate, i.e. all "RSA_* " cipher suites (see for details [1].) This suite of TLS is supported by standard web browsers and servers.

Setup. The protocol is illustrated in Fig. 1. We assume that in a setup stage, a (self-)certified public key pair $(sk_C, cert_C)$ has been correctly stored in the browser's credential store. Further, we assume that U and S share a password $pw \in \mathcal{D}$ and a corresponding identifier $w \in \mathcal{W}$.

Execution. The user U initiates the protocol by requesting a URL, addressing the server S. After the browser B has resolved the address, B initiates the handshake with S. B randomly chooses a client randomness r_C and a session id sid, it forwards these parameters to S. Then, S chooses randomly a server randomness r_C, and appends to the client's message r_C a certified

public key pk_S. Then, B randomly chooses a premaster secret PMK and encrypts the secret with the server's public key pk_S. The premaster secret PMK is used to derive the master secret MSK, using a pseudo-random function PRF instantiated by the client and server randomness and some string denoted as "1". The master secret MSK is used to derive the session keys (k_1, k_2). The browser also proves possession of its certified private key sk_c by signing the previously negotiated messages Sig_c. Finally, B confirms the session key generation, using a pseudo-random function that takes as input the master secret MSK, some string denoted as "1", and the digest value H_1 of all previous messages. Then, S verifies Sig_c, using the client's certified public key pk_c and ensures that it is connected to C. Analogously, S generates the master secret MSK in order to derive the same session keys (k_1, k_2). Finally, S confirms the negotiated session parameters, using a pseudo-random function PRF that takes as input the master key MSK, some string denoted as "1", and the digest of all previous messages H_2. The keys (k_1, k_2) are used in the following to secure the communication between client and server. To signal that also the user is connected to S, the server sends the easy-to-recognize identifier w. The advantage of the protocol is that the server identifies the browser and hence ensures that a secure channel to the client is established. The user is reduced to identify w, she has not to be aware of other security indicators. Upon receipt of w, the browser B renders the message w' and requests for authentication (e.g., enter password into form). If and only if, w' matches to w. Then U sends the password pw in order authenticate to S.

3.3 Short Analysis of Security

A polynomial-time adversary will be able to break the protocol by attempting to impersonate a user and making a brute-force attack against the server, trying to guess all passwords and trying to guess browser's private key. The adversary will also be able to break the protocol by attempting to impersonate a server and making a brute-force attack against the user, trying to guess all identifiers in order to perceive the password and trying to guess the browser's private key.

4 Conclusion

In this paper, we propose a browser-based protocol that makes the lowest assumptions on user's skills. In order to authenticate the server, the user has to detect a single identifier—be it an image or a passphrase. It remains an open questions, if indeed all users detect the identifier. For instance, what happens, when the user sees an identifier, which is homographic to the original. How can we then define a detect function? Our future work aims at elaborating a framework that allows a formal proof of the proposed protocol. This framework has to take into account the browser's protocol unawareness and should include the analysis of browser languages. Apart from a proof, an interesting aspect for future work is to model alternative human skills, such as the ability to solve human-hard puzzles. First steps towards those formal treatments have been recently made by Canetti, Halevi, and Steiner [3].

References

[1] C. Allen and T. Dierks. The TLS protocol — version 1.1. Internet proposed standard RFC 4346, 2006.

[2] Bresson, Chevassut, and Pointcheval. Security proofs for an efficient password-based key exchange. In *SIGSAC: 10th ACM Conference on Computer and Communications Security*. ACM SIGSAC, 2003.

[3] R. Canetti, S. Halevi, and M. Steiner. Mitigating dictionary attacks on password-protected local storage. In *CRYPTO*, pages 160–179, 2006.

[4] R. Dhamija and J. D. Tygar. The Battle Against Phishing: Dynamic Security Skins. In *SOUPS '05: Proceedings of the 2005 Symposium on Usable Privacy and Security*, pages 77–88, New York, NY, USA, 2005. ACM Press.

[5] R. Dhamija, J. D. Tygar, and M. A. Hearst. Why phishing works. In R. E. Grinter, T. Rodden, P. M. Aoki, E. Cutrell, R. Jeffries, and G. M. Olson, editors, *CHI*, pages 581–590. ACM, 2006.

[6] T. Groß. Security analysis of the SAML single sign-on browser/artifact profile. In *Proc. 19th Annual Computer Security Applications Conference*, 2003.

[7] T. Gross, B. Pfitzmann, and A.-R. Sadeghi. Browser model for security analysis of browser-based protocols. In *ESORICS*, 2005.

[8] Halevi and Krawczyk. Public-key cryptography and password protocols. *ACMTISS: ACM Transactions on Information and System Security*, 2, 1999.

[9] M. Jakobsson, S. Myers, and M. Augiere. Delayed Password Disclosure, 2005. `http://www.informatics.indiana.edu/markus/stealth-attacks.htm`.

[10] Katz, Ostrovsky, and Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 2001.

[11] D. Kormann and A. Rubin. Risks of the passport single signon protocol. *Computer Networks*, 33(1–6):51–58, 2000.

[12] R. Oppliger, S. Gajek, and R. Hauser. Security of mircrosoft's identity metasystem and cardspace. In *Kommunikation in Verteilten Systemen*, 2007.

[13] R. Oppliger, R. Hauser, and D. Basin. SSL/TLS Session-Aware User Authentication-Or How to Effectively Thwart the Man-in-the-Middle, 2005. (Computer Communications, accepted for publication).

[14] C. Soghoian and M. Jakobsson. A deceit-augmented man in the middle attack against bank of america's sitekey service, 2007. http://paranoia.dubfire.net/2007/04/deceit-augmented-man-in-middle-attack.html.

[15] M. Steiner, P. Buhler, T. Eirich, and M. Waidner. Secure password-based cipher suite for TLS. 4(2):134–157, May 2001.

[16] A. O. Stuart Schechter, Rachna Dhamija and I. Fischer. The emperor's new security indicators, 2007. to appear in the Proceedings of the IEEE Symposium on Security and Privacy.

[17] X. Suo, Y. Zhu, and G. S. Owen. Graphical passwords: A survey. In *ACSAC*, pages 463–472, 2005.

A Novel Impossible Differential Cryptanalysis of AES

Behnam Bahrak and Mohammad Reza Aref

Information System and Security Lab. (ISSL), Electrical Engineering Department, Sharif University of Technology, Tehran, Iran
{bahrak,aref}@ee.sharif.edu

Abstract

In 2000, Biham and Keller presented an impossible differential cryptanalysis of the Advanced Encryption Standard (AES) up to 5 rounds. This was improved in 2001 by Cheon et al. to apply to 6 rounds of the AES. In 2004 Phan extended this attack on the AES up to 7 rounds. In this paper, we present a new impossible differential attack on AES-128 reduced to 7 rounds which requires $2^{117.5}$ chosen plaintexts and 2^{109} bytes of memory and performs 2^{123} 7-round AES encryptions.

Keywords. Impossible Differential Cryptanalysis, Advanced Encryption Standard

1 Introduction

The Advanced Encryption Standard (AES) is a 128-bit block cipher with a secret key of length 128 , 192 or 256 bits. A 128-bit data block of the AES is usually exhibited as an array of 44 bytes with byte indexing as shown in Figure 1.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Figure 1: Byte coordinate of 128-bit data block

The AES has 10, 12 or 14 rounds, depending on the key size. Each round except the last, consists of 4 transformation, ByteSubstitution, a nonlinear 8×8 S-box byte- wise substitution; ShiftRow, a cyclic shift of each row with different offsets; MixColumn, a linear transformation of the 4 bytes in a column; and AddRoundKey, an XOR of the data block with round key. Before the first round AddRoundKey is applied and in the last round MixColumn is excluded [DR02]. In 2000, Biham and Keller [BK00] presented an impossible differential cryptanalysis of the Advanced Encryption Standard (AES) up to 5 rounds. This was improved in 2001 by Cheon et al. [CK^+01] to apply to 6 rounds of the AES. In 2004 Phan [Ph04] extended this attack on the AES up to 7 rounds. In this paper we present a new impossible differential attack on AES-128 reduced to 7 rounds which requires $2^{117.5}$ chosen plaintexts and 2^{109} bytes of memory and performs 2^{123} 7-round AES encryptions.

2 Four round impossible differential property of AES

In [BK00], a four-round impossible differential property was presented. This 4-round property states that given a pair of plaintexts which are equal in all bytes except one in which the pair differs, then the ciphertexts after 4 rounds cannot be equal in any of the following combinations of byte positions: (1,8,11,14), (2,5,12,15), (3,6,9,16) nor (4,7,10,13). In [BK00,CK^+01,Ph04], this property was used to attack the reduced round AES. Here we state a new impossible differential property which our attack bases on it. The new impossible differential property states that given a pair of plaintexts which are equal in all bytes except one, then the ciphertexts after 4 rounds cannot be equal in all bytes except 3 bytes in one column in which the pair differs. Figure 2 illustrates this property in one of possible cases. The boxes with black circle refer to

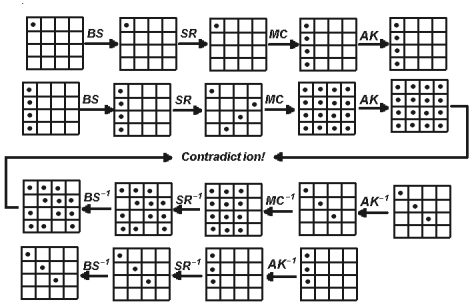


Figure 2: New impossible differential property of AES

the bytes in which the pair differs while the white boxes denote the equal bytes in the pair. Arrows labeled BS, SR, MC and AK , denote the ByteSubstitution, ShiftRow, MixColumn and AddRoundKey operations respectively and arrows labeled BS^{-1}, SR^{-1}, MC^{-1} and AK^{-1}, denote the inverse of the operations.

3 New impossible differential attack on 7 rounds of AES

In this section, we present an attack on 7 rounds of AES using the new impossible differential property which was described in previous section. The procedure of this attack is as follow:

1. A structure is defined as a set of 2^{32} plaintexts which have fixed values in all but four bytes (1,6,11,16). Such a structure proposes $2^{32} \times (2^{32}-1) \times \frac{1}{2} \approx 2^{63}$ pairs of plaintexts.

2. Take $2^{85.5}$ structures (i.e. $2^{85.5} \times 2^{32} = 2^{117.5}$ plaintexts, so $2^{85.5} \times 2^{63} = 2^{148.5}$ plaintext pairs). Select only the pairs whose ciphertext pairs after 7 rounds are equal in bytes (2,3,5,6,9,12,15,16). The expected number of such pairs is $2^{148.5} \times (2^{-8})^8 = 2^{84.5}$.

3. Guess a 32-bit value at bytes (4,7,10,13) for the last round key RK7. We name this 32-bit value RK71. For each ciphertext pair (C, C^*) , compute $C_{61} = MC^{-1} \circ BS^{-1} \circ SR^{-1}(C \oplus RK71)$ and $C^*_{61} = MC^{-1} \circ BS^{-1} \circ SR^{-1}(C^* \oplus RK71)$ and choose pairs whose difference $(C_{61} \oplus C^*_{61})$ are zero at all bytes except byte 8. The probability of such a difference is $(2^{-8})^3 = 2^{-24}$ and consequently we expect to have $2^{84.5} \times 2^{-24} = 2^{60.5}$ pairs with this condition.

4. Guess a 32-bit value at bytes (1,8,11,14) for the last round key RK7. We name this 32-bit value RK72. In figure 3, the black boxes refer to the bytes which we guessed the key

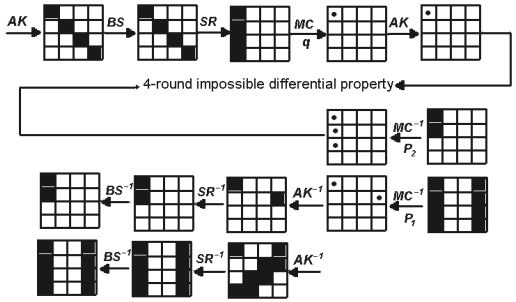


Figure 3: 7-round Impossible Differential Attack

for them and are known For each of $2^{60.5}$ remained ciphertext pairs (C, C^*), compute $C_{62} = BS^{-1} \circ SR^{-1}(C \oplus RK72)$ and $C^*_{62} = BS^{-1} \circ SR^{-1}(C^* \oplus RK72)$ and choose pairs whose difference $MC^{-1}(C_{62} \oplus C^*_{62})$ are zero at all bytes except byte 1.The probability of such a difference is $(2^{-8})^3 = 2^{-24}$, so we expect to have $2^{60.5} \times 2^{-24} = 2^{36.5}$ pairs with this condition. So at the end of this step, we have $2^{36.5}$ pairs which have zero difference in all bytes except bytes 1 and 8. We name these pairs (C_6, C^*_6).

5. Guess a 16-bit value at bytes (1,8) for the key $RK6^{'}(RK6^{'} = MC^{-1}(RK6)$. For $2^{36.5}$ pairs (C_6, C^*_6) compute the following difference $MC^{-1}([BS^{-1} \circ SR^{-1}(C^*_6 \oplus RK6^{'})] \oplus [BS^{-1} \circ SR^{-1}(C_6 \oplus RK6^{'})])$ and choose pairs whose differenc are zero at all bytes except three bytes in first column. The probability of such a difference is $2^{-8} \times 4 = 2^{-6}$, so we expect to have $2^{36.5} \times 2^{-6} = 2^{30.5}$ pairs which have such a difference. Note that $BS^{-1} \circ SR^{-1}(C^*_6 \oplus RK6^{'})$ and $BS^{-1} \circ SR^{-1}(C_6 \oplus RK6^{'})$ are 2 columns which are equal at third and fourth bytes, so XOR of them is zero at these bytes and we don't need to know the exact value of them in 3rd and 4th bytes.

6. In this part, we eliminate wrong 32-bit values at bytes (1,6,11,16) for the first round key RK0 by showing that the impossible property holds if these keys were used. We use the same precomputation as in [BK00]. At this precomputation stage we consider all of the pairs of the form $((a, b, c, d), (a^{'}, b, c, d))$ in a column (the data after the first round). For these pairs we perform $BS^{-1} \circ SR^{-1} \circ MC^{-1}$ and create a hash table containing one of the outputs of BS^{-1} (x) and the XOR of the two outputs $(x \oplus y)$. There are 2^{32} possible values for $(x \oplus y)$ and 2^{40} values for x, so on average there are about 2^8 values of x which correspond to each value of $(x \oplus y)$. Now for each of the $2^{30.5}$ reminded pairs we compute $(x \oplus y)$, and use the hash table fetch the about 2^8 possibilities of x which correspond to the computed $(x \oplus y)$. This process identifies about 2^8 wrong values for the key by XORing the plaintext and the x's. After analyzing all $2^{30.5}$ pairs, we expect only $2^{32} \times (1 - 2^{-32})^{2^{30.5} \cdot 2^8} \approx 2^{-99}$ wrong values of the 4 bytes of RK0 remain Unless the initial guess of the 64-bit value of the last round key RK7 and or the 16-bit value of the key $RK6^{'}$ is correct, it is expected that we can eliminate the whole 32-bit value of RK0 in this step since the wrong value (RK0, $RK6^{'}$, RK7) remains with the small probability of $(2^8)^{10} \times 2^{-99} = 2^{-19}$. Hence if there remains a value of RK0 , we can assume the guessed values for $RK6^{'}$ and RK7 are correct.

7. Step 3 requires $2 \times 2^{32} \times 2^{84.5} = 2^{117.5}$ one round operations. Step 4 requires $2 \times 2^{32} \times 2^{32} \times 2^{60.5} = 2^{125.5}$ one round operations. Step 5 requires $2 \times 2^{64} \times 2^{16} \times 2^{36.5} = 2^{117.5}$ one round operations. Step 7 requires $2^{64} \times 2^{16} \times 2^{30.5} \times 2^8 \approx 2^{118.5}$ one round operations. Consequently

the overall complexity of the attack is $\frac{2^{117.5}+2^{125.5}+2^{117.5}+2^{118.5}}{7} \approx 2^{123}$. Meanwhile, $\frac{2^{112}}{2^3} = 2^{109}$ bytes of memory are needed to store the list of deleted key values (RK0, $RK6'$, RK7).

4 Conclusion

We have presented a new impossible differential attack against AES-128 reduced to 7 rounds. The time complexity and required memory of this attack is much less than the previous impossible differential attacks [4]. In table 1, we compare the new attack with the previous impossible differential attacks. In table 2, we compare our attack with other attacks which applied on 7 rounds of AES-128.

Table 1: Comparison of impossible differential cryptanalysis of AES variants

Variant	**Rounds**	**Data**	**Workload**	**Memory**	**Source**
AES-128	5	$2^{29.5}$	2^{31}	2^{42}	[BK00]
AES-128	6	$2^{91.5}$	2^{122}	2^{93}	[CK^+01]
AES-128	7	$2^{117.5}$	2^{123}	2^{109}	This paper
AES-192	7	2^{92}	2^{186}	2^{157}	[Ph04]
AES-256	7	$2^{92.5}$	$2^{250.5}$	2^{157}	[Ph04]

Table 2: Comparison of our results with previous attacks on 7-round AES-128

Attack	**Data**	**Workload**	**Memory**	**Source**
Partial sum	$2^{128} - 2^{119}$	2^{120}	2^{61}	[FK^+01]
Collision	2^{32}	2^{128}	2^{80}	[GM00]
Impossible differential	$2^{117.5}$	2^{123}	2^{109}	This paper

References

[DR02] Joan Daemen and Vincent Rijmen. The Design of Rijndael. Information Security and Cryptography. Springer Verlag, 2002.

[BK00] E. Biham and N. Keller, 'Cryptanalysis of Reduced Variants of Rijndael', 3rd AES Conference, 2000

[CK^+01] J.H. Cheon, M. Kim, K. Kim, J.-Y. Lee, S. Kang, 'Improved impossible differential cryptanalysis of Rijndael and Crypton', in: Proc. 3rd International Conference on Information Security and Cryptology (ICISC 2001), in: Lecture Notes in Comput. Sci., vol. 2288, Springer-Verlag, Berlin, pp. 39-49, 2001.

[Ph04] Raphael Chung-Wei Phan, 'Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standard (AES)', Information Processing Letters, Vol. 91, Number 1, pp. 33-38, Elsevier, 2004.

[FK^+01] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, 'Improved cryptanalysis of Rijndael', in Fast Software Encryption, FSE 2000 (B. Schneier, ed.), vol. 1978 of Lecture Notes in Computer Science, pp. 213-230, Springer-Verlag, 2001.

[GM00] H.Gilbert and M.Minier., 'A collision attack on 7 rounds of Rijndael',3 Advanced Encryption Standard Candidate Conference, National Institute of Standards and Technology, pp. 230-241, April 2000.

Design Criteria for Key Mixture Functions of Block Ciphers

M. Tayarani Najaran[1], T. Eghlidos[2] and M. R. Aref[3]

[1] ISSL Lab, School of Electrical Engineering
Sharif University of Technology, Tehran, Iran
tayarani@mehr.sharif.edu

[2] Electronics Research Center
Sharif University of Technology, Tehran, Iran
teghlidos@sharif.edu

[3] ISSL Lab, School of Electrical Engineering
Sharif University of Technology, Tehran, Iran
aref@sharif.edu

Abstract

The design of symmetric key block ciphers has evolved throughout the years, after the invention of the first standard, DES. Various parts of this cipher system can be more or less found unchanged in systems designed after it. One of those parts is the key mixture function.

Designers have always tried to immunize their designed algorithms against generic attacks. Most of the attention for this purpose has been paid to non-linear parts. In this paper we focus on another important part of an encryption algorithm, the key mixture function, and present five design criteria in order to enable strengthening encryption algorithms against various methods of cryptanalysis. We also propose methods to evaluate these properties numerically.

Keywords. Block Ciphers, Differential Cryptanalysis, Modular Arithmetic

1 Introduction

One type of symmetric-key encryption systems are block ciphers, in which a fixed-size block of data is substituted with a block of encrypted cipher. The cipher should have a deterministic relation with the plaintext and an encryption key via the encryption algorithm. Data secrecy is required to be based on the secrecy of the key. Due to the deterministic nature of the encryption algorithm, knowledge of the ciphertext may sometimes jeopardize data secrecy. Hence, a fixed number of 'sub-key' bits are extracted from the encryption key and mixed with intermediate values to conceal data patterns. This mixture is done using a 'key mixture' function.

Many generic methods of cryptanalysis extract information about the plaintext and/or the encryption key by taking advantage of data leakages in the encryption algorithms. Less attention has been paid to the effect key mixture functions may have on the security of the system. In this paper, we try to emphasize on the importance of such functions and provide design criteria to help find possible better key mixture functions.

The structure of this paper is as follows: Section 2 introduces two common types of key mixture functions mainly used in secure encryption algorithms. The strength and weaknesses of each type are briefly explained. Section 3 is a short explanation of a few generic methods of cryptanalysis, which were the main source that the criteria were derived from and take advantage

of weak key mixture functions. The proposed design criteria are presented in Section 4. Further research guidelines and conclusions can be found in Sections 5 and 6, respectively[1].

2 Common Key Mixture Functions

Many encryption algorithms, such as DES [DES] and AES [AES], use the binary exclusive-Or (Xor) function as the key mixture function. The Xor function has easy hardware implementation and fast software execution. Its most important property is that the number of 1's and 0's are equal in its output truth table. The main weakness of the Xor function is that $x \bigoplus x = 0$.

Many other secure encryption algorithms, such as IDEA [IDEA] and RC6 [RiRo99+], use modular arithmetic for mixing data bits with key bits. Modular multiplication provides much better security, compared to the Xor function, but the main reason why modular multiplication is not widely used is because of its high time consumption and difficult implementation. In the case where good performance is also required, the block size may not be freely chosen.

3 Generic Cryptanalysis Methods

3.1 Differential Cryptanalysis

Differential cryptanalysis [BiSh90] cancels out the effect of the round sub-key by Xor-ing two keyed values with each other. By expressing the S-boxes with probabilistic Xor relations, a probabilistic relation may be found that is independent of the values of the sub-keys and helps attackers find candidates for the last round sub-key in an exhaustive search manner.

3.2 Linear Cryptanalysis

Linear cryptanalysis, first proposed by Matsui [Mat93], uses a probabilistic linear relation to relate plaintexts bits, ciphertext bits and key bits using the Xor function. The linear expression is obtained by finding the correlations between the input bits and output bits of each S-boxes.

3.3 Algebraic Cryptanalysis

Algebraic cryptanalysis methods use deterministic algebraic relations to express the encryption algorithm. These methods have different ways to produce the algebraic equations. The generic cryptanalysis method known as the algebraic attack uses relations in GF(2), where Xor serves as addition. Solving this algebraic system leads to finding the encryption key [CoPi02].

3.4 Impossible Differentials Cryptanalysis

The impossible differentials cryptanalysis [BiKe00] takes advantage of impossible events that never occur in the system. The attack is carried out with an exhaustive search for certain bytes of some sub-keys, where wrong sub-keys are discarded if decrypting the last round with this sub-key candidate results in an impossible event happening. It can be shown that the probability of a wrong key to pass this test is very small.

Most of the methods presented here use *differentials* as a basis to the attack and the fact that the Xor function is weak against differentials. The rest of the methods use the Xor function as bitwise addition.

[1]This paper is partially supported by Iran Telecommunication Research Center (ITRC).

4 Design Criteria

A key mixture function may or may not be reversible and so, reversibility is not considered as a design criterion. In the rest of this section, denote f the key mixture function.

4.1 Balance

The first requisite of any key mixture function is Balance.

DEFINITION 4.1 *: The function f is said to be balanced if it takes all possible output values with equal probability.*

Unbalanced key mixture functions inject bias into the output data patterns. This is a source of information leakage, since the output probability distribution can provide information about the key. Criterion 1 helps measure this bias.

Criterion 1: A good key mixture function is expected to have an output with a uniform probability distribution function. In other words, C_1 defined by Equation 1 is supposed to be close to 1 as possible. In an ideal state we want C_1=1.

$$C_1 \triangleq 1 - \Sigma_{i=1}^{N} |p_i - \frac{1}{N}| \tag{1}$$

Where N is the total number of possible outputs for all possible inputs of the function f, and p_i is the probability of the i-th output occurring.

Balance may be interpreted as the statistical deviation of the output of the key mixture function. The major difference is that we used the absolute value function instead of the square function to increase the effect of not being balanced on the value of C_1. However, we emphasize that if necessary, the definition of C1 may be altered.

The Xor function completely satisfies this criterion which is one of its very important properties. Modular multiplication, on the other hand, cannot completely satisfy this criterion.

4.2 Coverage

Coverage of the key-mixture function is very close in meaning to Balance, but we have intentionally chosen to distinguish between them.

DEFINITION 4.2 *: Coverage of the function f is the ratio of the number of outputs of the function f with non-zero probabilities to all possible outputs.*

Bad coverage of the function f injects bias into the system and can be considered as a special case of an unbalanced function. We chose to separate Balance and Coverage to distinguish between a balanced function with bad coverage and an unbalanced function with full coverage. Each one can be compromising depending on the structural design. Choosing which type to use is left to the designer.

Criterion 2: A good key mixture function is expected to have the most coverage as possible. In other words, C_2 is supposed to be close to 1 as possible. Once again, in an ideal state we want C_2=1. C_2 is defined as follows, where m is the output bit length of the function f:

$$C_2 \triangleq \frac{\#possible\ outputs}{2^m} \tag{2}$$

The Xor function has complete coverage. The modular multiplication function has some sort of a trade-off between balance and coverage. If the modus is prime, balance is better at the price of low coverage, due to the fact that outputs greater than the modus never occur. If good coverage is required by choosing a non-prime modus of 2^n, balance is lost drastically.

4.3 Data/Key Ratio

Sub-key bits are considered as one of the resources of the system. It can be deduced that using more sub-key bits can help increase the security of the algorithm. So, with equal security, the less sub-key bits used by a key mixture function, the better it is, whereas, decreasing the number of total sub-key bits decreases the effective key search space. Overall, there should be a balance between the number of data and sub-key bits mixed together. Also, all output bits should be dependent on the sub-key bits. Criterion 3 combines all these conditions together.

Criterion 3: For a fixed number of data bits, a good key mixture function is expected to use as less sub-key bits as possible, while the overall number of sub-key bits is still sufficiently large and all output bits depend on input sub-key bits. C_3, defined below, is supposed to be close to 1 as possible and equal to 1 in an ideal state, presuming the other two conditions are satisfied.

$$C_3 \triangleq max(0, 1 - \frac{input\ subkey\ length}{input\ data\ length}) \tag{3}$$

4.4 Transparency

Some methods of cryptanalysis have taken advantage of the fact that Xor-ing a value with itself is zero to cancel the effect of key mixture. Satisfying Criterion 4 helps improve security against these methods of cryptanalysis.

DEFINITION 4.3 *: The function f(a,b) is said to be strongly transparent to g regarding b if:*

$$\forall b : g(f(a,b), f(c,b)) = h(a,c) \tag{4}$$

Where a and c may have any value. If a and c should have a specific relation, it is said to be weakly transparent to g regarding b.

The Xor function is strongly transparent to Xor regarding both its inputs. Modular multiplication, on the other hand, is weakly transparent to modular addition regarding b. Criterion 4 evaluates this property. Unlike the other three, this criterion only takes three different values.

Criterion 4: A good key mixture function is required to be neither weakly nor strongly transparent to any function regarding the sub-key k_i. C_4 may take the following three different values:

$$C_4 \triangleq \begin{cases} 1 & \text{if f is not Transparent} \\ \frac{1}{2} & \text{if f is weakly Transparent} \\ 0 & \text{if f is strongly Transparent} \end{cases} \tag{5}$$

4.5 Complexity

The performance of the key mixture function has a great impact on the overall performance of the encryption algorithm. When designing a key mixture function, both software and hardware implementation efficiency should be taken into consideration.

Criterion 5: A good key mixture function is required to have the least implementation complexity as possible. This complexity can be either software or hardware benchmarks.

The last criterion may be evaluated numerically, but it is not possible to normalize its value to the interval [0,1]. The Xor function is by far much more efficient than modular arithmetic, both in hardware and software implementation. As far as implementation complexity is required to be as low as possible, the algebraic terms representing the function are desired to be complex.

4.6 Putting It All Together

The required criteria are combined by a linear expression in the form of Equation 6. For the linear combination, the weights were given in order of importance: Transparency, Balance, Coverage and Data/Key Ratio with values 0.4, 0.3, 0.2 and 0.1, respectively.

$$C = \alpha \times C_1 + \beta \times C_2 + \gamma \times C_3 + \delta \times C_4 \quad (6)$$

$$\alpha + \beta + \gamma + \delta = 1,\ \alpha, \beta, \gamma, \delta \geq 0 \quad (7)$$

Using these weights the Xor function and the modular multiplication function X×k (mod n) have been compared with each other for different values of n. The results are listed in Table 1. The used values for n are 2^{16}+1, as used in IDEA, 2^{16} to show the trade-off between balance and coverage and 2^{16}-15 which is the largest 16-bit prime number. Note that in calculating C, performance was not taken into consideration. The last column of Table 1 shows the performance differences, measured with a software execution time metric only for easy comparison.

Table 1: Evaluated Values Using Equation 6 Function

Function	$\mathbf{C_1}$	$\mathbf{C_2}$	$\mathbf{C_3}$	$\mathbf{C_4}$	**C**	**Clock**
Xor	1.0000	1.0000	0.0	0.0	0.5000	1
n= 2^{16}+1	0.9999	0.5000	0.0	0.5	0.5999	1006
n= 2^{16}	0.5000	1.0000	0.0	0.5	0.5500	57
n= 2^{16}-15	1.0000	0.9998	0.0	0.5	0.6999	970

5 Further Research

The proposed criteria help designers numerically evaluate various key mixture functions. We invite designers to propose new functions best satisfying the design criteria to improve security.

6 Conclusions

In this paper we emphasized on the key mixture function. Common used key mixture functions were mentioned, followed by a general description of cryptanalysis methods taking advantage of key mixture weaknesses. We proposed five design criteria as minimum requisites for any key mixture function. The proposed quantitative measures of the criteria help compare key mixture functions. The values can be combined by an expression dependent on design intentions.

References

[AES] J. Daemen, V. Rijmen, "AES Proposal: Rijndael (corrected version)", 2nd AES Conference, http://csrc.nist.gov/CryptoToolkit/aes/round2/round2.htm, 1999.

[BiKe00] E. Biham, N. Keller, "Cryptanalysis of Reduced Variants of Rijndael", http://csrc.nist.gov/encryption /aes/round2/conf3/aes3papers.html.

[BiSh90] E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", Advances in Cryptology, Springer-Verlag, Proceedings of Crypto'90, pp2-21, 1990.

[CoPi02] N. T. Courtois, J. Pieprzyk, "Crypanalysis of Block Ciphers with Over Defined Equations", Asiacrypt'02, Springer Verlag, pp267-287, 2002.

[DES] "Data Encryption Standard", FIPS PUB 46-2, www.itl.nist.gov/fipspubs/fip46-2.htm.

[IDEA] IDEA, www.mediacrypt.com.

[Mat93] M. Matsui, "Linear Cryptanalysis Method for DES Cipher", Advances in Cryptology (LNCS), Springer-Verlag, Proceedings of Eurocrypt'93, pp386-397, 1993.

[RiRo99+] R. L. Rivest, M. J. B. Robshaw, R. Sidney, Y. L. Lin, "The RC6 Block Cipher", 2nd AES Conference, http://csrc.nist.gov/CryptoToolkit/aes/ round2/round2.htm, 1999.

New Key Schedule Prototype With Provable Security

M. Tayarani Najaran[1], T. Eghlidos[2] and M. R. Aref[1]

[1] ISSL Lab, School of Electrical Engineering
Sharif University of Technology, Tehran, Iran
[2] Electronics Research Center
Sharif University of Technology, Tehran, Iran
tayarani@mehr.sharif.edu, teghlidos@sharif.edu, aref@sharif.edu

Abstract

In this paper we propose a new key schedule prototype that provides provable security against several methods of cryptanalysis, including Related Key attacks, and also improves security against other types of cryptanalysis, such as differential attacks and exhaustive search, by preserving the effective key-search space of the main key even with knowledge of all sub-keys.

Keywords. Key schedule attacks, Differential attacks, One-way function, Hash function

1 Introduction

The main key of an encryption algorithm is expanded into sub-key bits using a key schedule, where each sub-key is normally fed into the round function used for encryption. Many types of cryptanalysis apply on the round function revealing part or all of the corresponding sub-key.

In this paper we propose a new prototype for the key schedule of a block cipher. In Section 2, two different types of cryptanalysis are explained and in Section 3 the prototype is proposed, along with the immunity criteria required. Section 4 is a conclusion of what we presented.

2 Cryptanalysis

There are two kinds of cryptanalysis methods somehow dealing with the key schedule. The first group directly exploit weaknesses of the key schedule. Naming a few would be: Related Key attack, where the relation between two keys reveals their value; Slide attack, where a new set of sub-keys are produced with a one-round shift to the first set; weak keys, which lead to equal plaintexts and ciphertexts; Detectable key classes, which with small workload, it can be determined whether or not the used key belongs to a smaller space[KeScWa96].

The other cryptanalytic methods of our intention aim at the round function itself and somehow reveal certain bits of the sub-keys. After revealing the sub-key bits, the key schedule is worked back to find the corresponding main key bits. This reduces the effective key search space of the main key [BiSh92].

3 The Proposed Prototype

Out proposed prototype satisfies three major design criteria defined below:

Criterion 1: A good key schedule is required to be relation-resistant, i.e the probability that two different sets of sub-keys produced by two different keys having any specific relation should be the least as possible and zero in an ideal case.

Criterion 2: A good key schedule is required to be one-way, i.e given any subset of sub-key bits the number of revealed bits from the main key should be the least as possible.

Criterion 3: A good key schedule is required to have the highest possible setup time.

Ideally satisfying Criterion 1 provides immunity against key schedule attacks, such as Related Key attack, since no special relation can be derived by changing the main key.

Ideally satisfying Criterion 2, i.e. infeasibility of finding any main key bits even by having all sub-key bits, increases immunity against all generic methods of cryptanalysis that obtain a subset of sub-key bits. This is because the rest of the sub-key bits should be found in an exhaustive search in order to compromise data secrecy. Considering the facts that normally the number of sub-key bits are much more than the main key size and most methods of cryptanalysis only reveal a few bits of the sub-keys, the remaining exhaustive search space is much greater than the original exhaustive search space of the main key.

Satisfying Criterion 3 helps resisting an exhaustive search attack, since producing the sub-keys is off-line, increasing the key setup time by a factor t has negligible effect on overall encryption time, whereas the time required for an exhaustive search is multiplied by t [QuDeDa85].

The proposed scheme is depicted in Figure 1. The two blocks Alg1 and Alg2 can be freely designed. The hash function can also be replaced with any one-way function. Since the hash

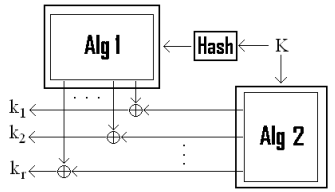


Figure 1: New Key Schedule Prototype

function is one-way, Criterion 1 is ideally satisfied. Also, even obtaining all sub-keys gives no information about the main key, because it is computationally infeasible to predict the relation between the main key and its hash value, and hence it ideally satisfies Criterion 2. This one-wayness also provides immunity against other key schedule weaknesses and attacks, such as Weak Keys, Equivalent Keys, Linear Factors and Meet-in-the-Middle attack. Additionally, the hash function increases the required setup time and so satisfies Criterion 3.

4 Conclusions

In this paper we presented a new prototype for the key schedule of a block cipher. The new prototype has provable resistibility against many types of cryptanalysis attacking the key schedule. Additionally, this prototype improves resistibility of the encryption algorithm against methods of cryptanalysis which reveal sub-keys and also against the exhaustive search attack.

References

[BiSh92] E. Biham, A. Shamir, "Differential Cryptanalysis of the Full 16-round DES Cryptosystem", Advances in Cryptology, Crypto 92, pp487-496, 1993.

[KeScWa96] J. Kelsey, B. Schneier, D. Wagner, "Key Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER and Triple-DES", Advances in Cryptology, Crypto 96, pp237-251, 1996.

[QuDeDa85] J. Quisquater, Y. Desmedt, M. Davio, "The Importance of Good Key Scheduling Schemes", Advances in Cryptology, Crypto 85, pp537-542, 1985.

Index

Aguilar-Melchor, Carlos, 50
Aref, Mohammad Reza, 55, 119, 152, 157, 163

Bahrak, Behnam, 152
Bechir, Ayeb, 61
Beiter, Michael, 115
Benits, Waldyr, 6

Courtois, Nicolas, 88

Dallot, Leonard, 66
Debraize, Blandine, 88
Ding, Jintai, 68

Eghlidos, Taraneh, 55, 119, 157, 163
Ehdaie, Mohammad, 119
Eisenbarth, Thomas, 11
Ekberg, Jan-Erik, 97

Fan, Junfeng, 79
Fleischmann, Ewan, 103

Güneysu, Tim, 31
Gaborit, Philippe, 50
Gajek, Sebastian, 147
Galbraith, Steven, 6
Gendrullis, Timo, 72
Gierlichs, Benedikt, 1

Hagui, Mabrouka, 61
Hartung, Rupert J., 45
He, Wei-Hua, 82
Holbl, Marko, 21

Imai, Hideki, 1
Indesteege, Sebastiaan, 86

Janussen, Kåre, 1
Juan, Fei-Ming, 82

Kasper, Timo, 72
Khader, Dalia, 108
Khammari, Hédi, 61
Korsten, Anja, 74
Kubwalo, Paul, 90

Lemke-Rust, Kerstin, 11
Liao, Lijun, 141

Manuel, Stephane, 26
Manulis, Mark, 147
Meiser, Gordon, 11
Mulagha, John, 90

Naya-Plasencia, María, 136

Otsuka, Akira, 1

Paar, Christof, 11, 31, 72
Paul, Souradyuti, 124
Piramuthu, Selwyn, 41
Plessers, Joris, 70
Preneel, Bart, 86, 124
Puttmann, Christoph, 16

Rechberger, Christian, 21
Röck, Andrea, 130

Sadeghi, Ahmad-Reza, 147
Sakiyama, Kazuo, 79
Schäege, Sven, 31
Schaefer, Edward F., 90
Schmidt, Dieter, 68
Schwarzpaul, Thomas, 36
Schwenk, Joerg, 141
Schwenk, Jörg, 147
Sebastian, Gajek, 141
Sekar, Gautham, 124
Sendrier, Nicolas, 26
Shigetomi, Rie, 1
Shokrollahi, Jamshid, 16
Sobhi Afshar, Ali Akbar, 55
Stegemann, Dirk, 84

Tayarani Najaran, Mahdi, 157, 163
Tu, Yu-Ju, 41

Vanderheyden, Caroline, 79
Verbauwhede, Ingrid, 70, 79

Welzer, Tatjana, 21
Werner, Fabian, 68
Wernsdorf, Ralph, 76

Yoshida, Rei, 1
Yoshizoe, Kazuki, 1