Network Security (NetSec)

Summer 2015
Chapter 02: Crypto Applied to Networks
Module 01: Crypto Design/Choice Considerations, Pitfalls

Prof. Dr.-Ing. Matthias Hollick

Technische Universität Darmstadt
Secure Mobile Networking Lab - SEEMOO
Department of Computer Science
Center for Advanced Security Research Darmstadt - CASED

Mornewegstr. 32
D-64293 Darmstadt, Germany
Tel.+49 6151 16-70922, Fax. +49 6151 16-70921
http://seemoo.de or http://www.seemoo.tu-darmstadt.de

Learning Objectives

- Learn about selected pitfalls in applying cryptographic protocols
  - What can go wrong (what will go wrong)?
  - Emphasis on basic functionality of cryptography: authentication protocols
- Understand design trade-offs in choosing appropriate cryptographic mechanisms
  - How expensive is crypto in terms of memory
  - How expensive is crypto in terms of computation
  - How expensive is crypto in terms of bandwidth and energy consumption
- Discuss which algorithms match which networking requirements

Outline

- Which crypto to choose?
- Symmetric encryption & public-key encryption in two slides
- Crypto handshakes & pitfalls
- Performance considerations in communication networks
- Performance considerations in wireless sensor networks

Which Crypto to Choose?

Which Crypto to Choose

Shannon's Guide to Good Ciphers

- Amount of secrecy should determine amount of labor appropriate for encryption and decryption
- The set of keys and enciphering algorithm should be free from complexity
- **The implementation should be as simple as possible**
- Errors in ciphering should not propagate
- **Size of the enciphered text should be no larger than the original**

Commercial Encryption Guides (Best Practice)

- Cryptosystem should be based on sound mathematics
- It has been analyzed by many experts
- It has stood the "test of time"

Secret-key Cryptography

Secret-key cryptography

- Same key to encrypt and decrypt message
- Sender sends message and key to receiver

Problems with secret-key cryptography

- Key must be **securely** transmitted to receiver
- Different key for every source-destination pair
- Key distribution centers (KDC) used to reduce these problems
  - Generates session key and sends it to nodes (sender, receiver) encrypted with a unique key between KDC and nodes

Encryption algorithms

- Data Encryption Standard (DES), Triple DES
- Advanced Encryption Standard (AES)

Public Key Cryptography

Public key cryptography

- Asymmetric – two inversely related keys (key pair)
  - Private key and public key
- If public key encrypts only private can decrypt and vice versa
- Each party has a key pair, i.e., both a public and a private key
- Can be used for encryption, signature, both

Problems with public key cryptography

- Requires computationally expensive operations

DSA, RSA or ECC are well known public key algorithms

- DSS/DSA: Digital Signature Standard, Digital Signature Algorithm
- RSA: Rivest, Shamir, Adleman
- ECC: Elliptic Curve Cryptography

Handshakes

-

A Crypto Classic

Cryptographic Handshakes

Let‘s assume Alice and Bob share the same symmetric key or have each other‘s public key

Once keys are known to two parties, need a handshake to authenticate

Goals:

- Mutual authentication
- Immune from replay and other attacks
- Minimize number of messages
- Establish a session key as a side effect

Notation for the Next Slides

Notation

- **f($K_{Alice-Bob}$, R)** or short **f(K,R)**
  means that R is cryptographically transformed with the shared secret $K_{Alice-Bob}$

- **K{R}** means that R is encrypted with K using AES or DES or alike
- **h{R}** or **hash{R}** means that R is hashed using K by producing a message digest or a concatenation of R and K

Protocol to Start With

Lots of LOGIN protocols have been designed for environments

- where eavesdropping was not a concern (rightly or wrongly), and
- where attackers were not sophisticated (rightly or wrongly)

Authentication in such a case could consist of

- Alice sending her name and password in clear
- Bob checks name and password and starts to communicate (obviously unencrypted, integrity not protected)

Alice

Bob

I'm Alice, K →

verifies Alice, K

Challenge/Response vs. Timestamp

What does the following protocol achieve?

Alice

Bob

I'm Alice →

← *a challenge* R

compare:

$f(K_{Alice\text{-}Bob}, R)$ →

#1

Bob authenticates Alice based on a shared secret K

Is it any better than the cleartext example?

What are weaknesses?

Challenge/Response vs. Timestamp

Which problems do you see in the following protocol #2?

Alice

Bob

I'm Alice →

← *a challenge* R

compare: $f(K_{Alice\text{-}Bob}, R)$ →

#1

I'm Alice →

← $K_{Alice\text{-}Bob}\{R\}$

compare: R →

#2

Challenge/Response vs. Timestamp

Which problems do you see in the following protocols?

Alice

Bob

I'm Alice →

a challenge R ←

compare: $f(K_{Alice\text{-}Bob}, R)$ →

#1

---

vs: I'm Alice, $K_{Alice\text{-}Bob}\{timestamp\}$ →

#3

Pitfalls with One-way Public Key

In the protocols with secret key, Trudy can impersonate Alice, if she gets access to the password database at Bob

Let's move to public key

Alice

Bob

I'm Alice →

← R

$[R]_{Alice}$ →

#4

Bob authenticates Alice based on her public key signature

What kind of protection is required for Bob's password database?

Pitfalls with One-way Public Key

And a variant similar to #2

Alice

Bob

I'm Alice →

← $\{R\}_{Alice}$

R →

#5

Bob authenticates Alice if she can decrypt a message encrypted with her public key

What are weaknesses of #4 and #5?

- #4 might trick Alice into signing something
- #5 might trick Alice into possibly decrypting something

Take Away from #4 and #5

Do not use the same key for different purposes

- Unless the design for all uses of the key are coordinated so that an attacker cannot use one of the protocols to help break another

- Coordination could be:
  - Give structure to R (such that you cannot mistake a challenge with let`s say an email)

Important

- You can design several schemes, each of which is independently secure, but using more than one of these can spell trouble!
- Deploying a new protocol using the same key in inappropriate fashion can compromise the security of existing schemes!

Mutual Authentication

Alice

Bob

I'm Alice →

← R1

#6

$f(K_{Alice\text{-}Bob}, R1)$ →

R2 →

← $f(K_{Alice\text{-}Bob}, R2)$

Mutual authentication based on shared secret K

Goal: Alice wants to be sure to communicate with Bob

More Efficient Mutual Authentication

#6 is inefficient (5 messages), so we put more than one item of information into a message

Alice

Bob

I'm Alice, R2 →

← R1, $f(K_{Alice\text{-}Bob}, R2)$

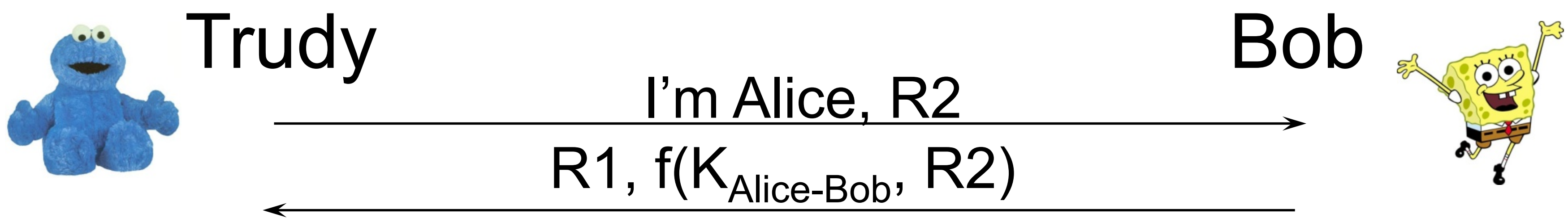
$f(K_{Alice\text{-}Bob}, R1)$ →

#7

What are weaknesses of #7?

#7 allows a reflection attack! See next slide!

Reflection Attack



Trudy starts a second parallel connection

Trudy completes the first

Secret Key Mutual Authentication

How does Server know Alice's Secret Key (implementation issues, making things scalable)?

Various possibilities:

- individually configured into each server
- authentication storage node (servers retrieve it from there)
- authentication facilitator node (does the authentication and answers yes/no)
- hopefully doesn't store password or password-equivalent

#7 is further prone to offline password guessing attacks!

Note that #6 did not suffer the problem of the reflection attack of #7!

Public Key Mutual Authentication

Alice

Bob

I'm Alice, $\{R2\}_{Bob}$ →

← R2, $\{R1\}_{Alice}$

R1 →

#8

R1 and R2 are encrypted with the public key of Alice and Bob, respectively

Need to ensure that public keys are correct!

Timestamp Based Mutual Authentication

Alice

Bob

I'm Alice, f{$K_{Alice\text{-}Bob}$,timestamp} →

I'm Bob, f{$K_{Alice\text{-}Bob}$,timestamp} ←

#9

Two messages instead of three
Must assure Bob's timestamp is different!

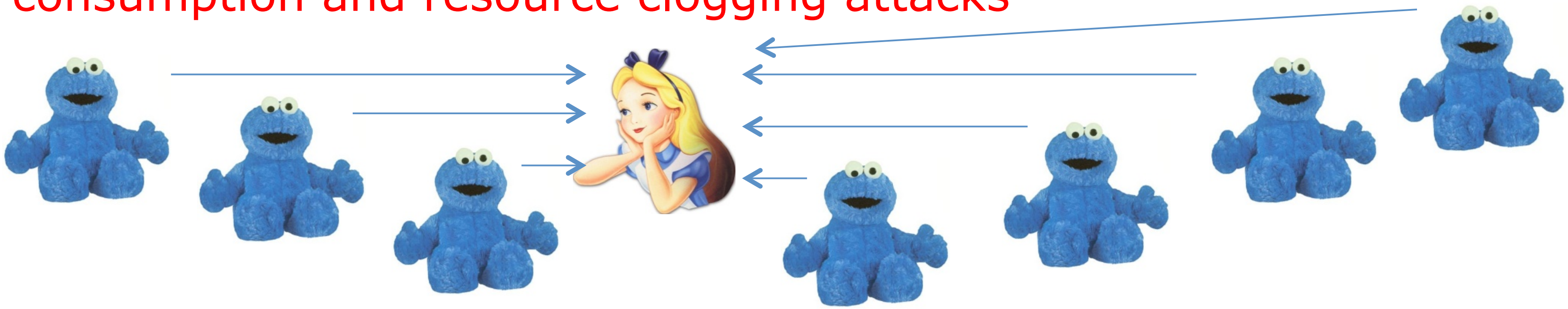
How to Protect Availability with Crypto (or other Mechanisms)

So far, we discussed pitfalls using basic cryptographic functions

Can we protect against attacks against the availabaility of the system using cryptography

- Particularly keeping in mind that in some cases, cryptography makes attacks against the availability of a system easier!

→ We will start a discussion thread in the forum, where you can propose and discuss mechanisms to protect against resource consumption and resource clogging attacks



Take Away Message

Keep always in mind: a minor variant of a secure protocol can be insecure

Keep always in mind: a secure protocol might get insecure if moved to a different environment with different assumptions, since a weakness suddenly gets exploitable

Acks & Recommended Reading

Selected slides of this chapter courtesy of

- Radia Perlman

Recommended reading

- Chapter 11 of
  [KaPeSp2002] Charlie Kaufman, Radia Perlman, Mike Speciner: Network Security – Private Communication in a Public World, 2nd Edition, Prentice Hall, 2002, ISBN: 978-0-13-046019-6

Performance Considerations

Performance of Crypto in Networks (here IPSec/SSL/SSH)

SECURE MOBILE NETWORKING

Crypto is in use in various protocols that we will discuss during the course of this lecture

- Which performance can be achieved with state-of-the-art crypto?

- In the following, we will give some insights into the performance of popular crypto algorithms applied in networks

- Sources:

A Study of the Relative Costs of Network Security Protocols*

Stefan Miltchev
miltchev@dsl.cis.upenn.edu
University of Pennsylvania

Sotiris Ioannidis
sotiris@dsl.cis.upenn.edu
University of Pennsylvania

Angelos D. Keromytis
angelos@cs.columbia.edu
Columbia University

Analyzing the Energy Consumption of Security Protocols

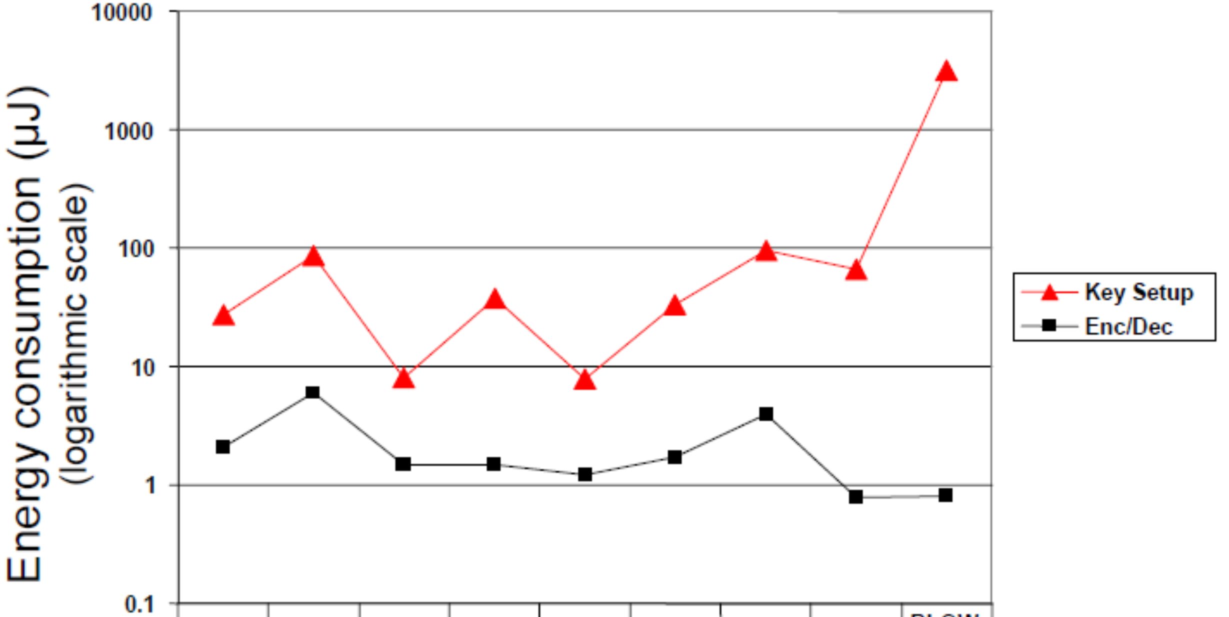
Nachiketh R. Potlapally[†], Srivaths Ravi[‡], Anand Raghunathan[‡] and Niraj K. Jha[†]
[†] Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544
[‡]NEC Laboratories America, Princeton, NJ 08540

Energy Costs of Symmetric Key Algorithms

SECURE MOBILE NETWORKING

Analyzing the Energy Consumption of Security Protocols

Nachiketh R. Potlapally†, Srivaths Ravi‡, Anand Raghunathan‡ and Niraj K. Jha†
† Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544
‡NEC Laboratories America, Princeton, NJ 08540



	DES	3DES	IDEA	CAST	AES	RC2	RC4	RC5	BLOW FISH	
Key Setup	27.53	87.04	7.96	37.63	7.87	32.94	95.97	66.54	3166.3	(μJ)
Enc/Dec	2.08	6.04	1.47	1.47	1.21	1.73	3.93	0.79	0.81	(μJ/byte)

Figure 3: Energy consumption data for various symmetric ciphers

Energy Costs of Hash, Signature and Key Exchange Algorithms

Hashes

Table 1: Energy consumption characteristics of hash functions

Algorithm	MD2	MD4	MD5	SHA	SHA1	HMAC
Energy (μJ/B)	4.12	0.52	0.59	0.75	0.76	1.16

Signature algorithms

Table 2: Energy cost of digital signature algorithms

Algorithm	Key size bits	Key generation (mJ)	Sign (mJ)	Verify (mJ)
RSA	1024	270.13	546.5	15.97
DSA	1024	293.20	313.6	338.02
ECDSA	163	226.65	134.2	196.23

Key exchange algorithms

Table 3: Energy cost of key exchange algorithms

Algorithm	Key size ($bits$)	Key generation (mJ)	Key exchange (mJ)
DH	1024	875.96	1046.5
ECDH	163	276.70	163.5
DH	512	202.56	159.6

Packet Sizes Matter

Per packet transaction overhead can lead to prohibitive costs of crypto mechanisms

Analyzing the Energy Consumption of Security Protocols

Nachiketh R. Potlapally†, Srivaths Ravi‡, Anand Raghunathan‡ and Niraj K. Jha†
† Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544
‡NEC Laboratories America, Princeton, NJ 08540

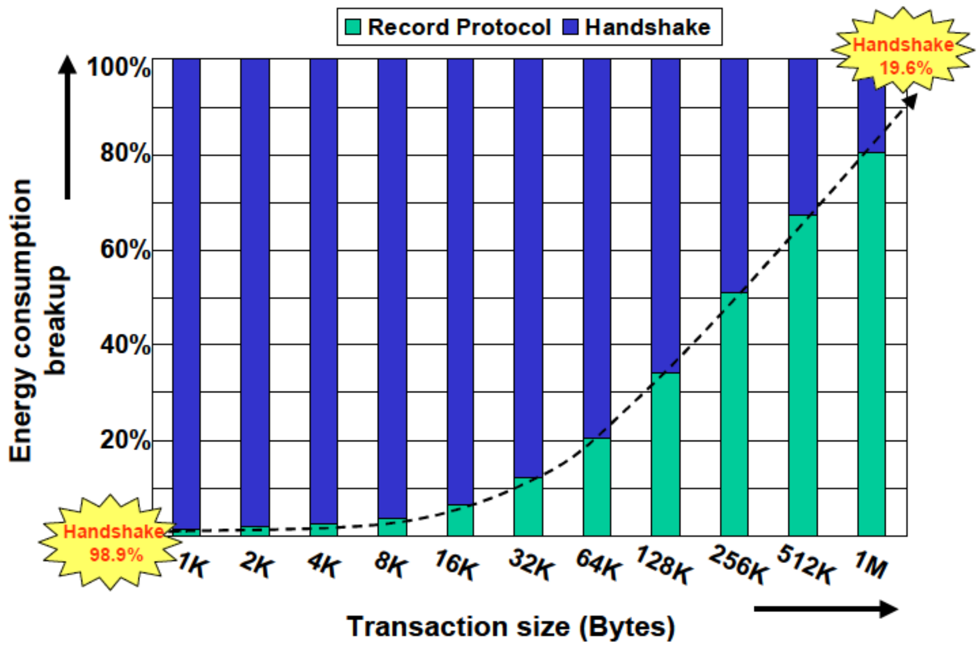


Figure 8: Variation of energy consumption contributions from SSL handshake and record stages with increasing transaction sizes

Asymmetry Between Client and Server

Depending on the chosen cipher, the workload for the server can be significantly higher than the workload of the client

- (Distributed) denial of service attacks can exploit this assymetry

Analyzing the Energy Consumption of Security Protocols

Nachiketh R. Potlapally†, Srivaths Ravi‡, Anand Raghunathan‡ and Niraj K. Jha†
† Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544
‡NEC Laboratories America, Princeton, NJ 08540

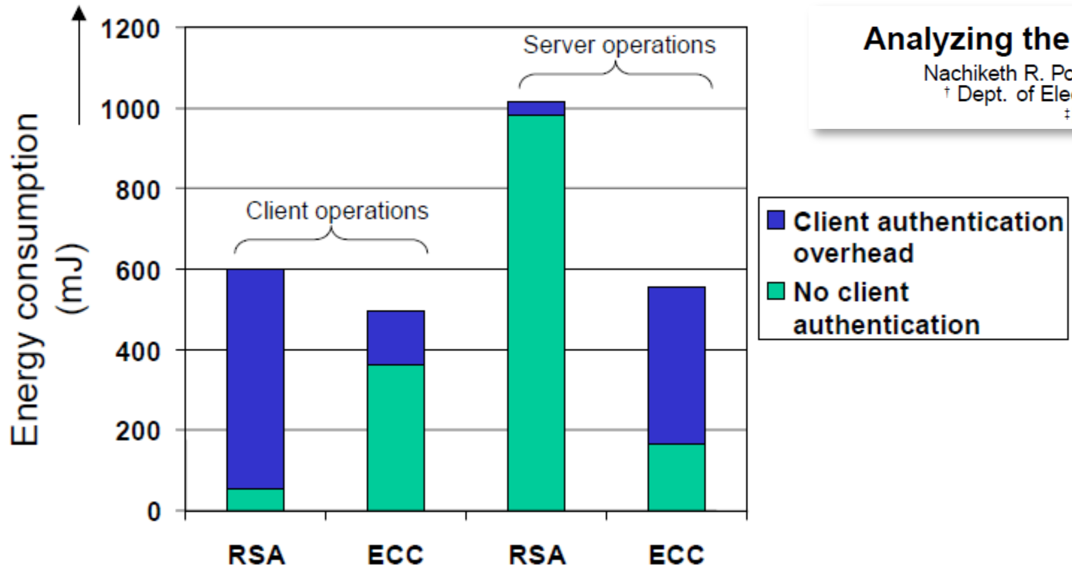
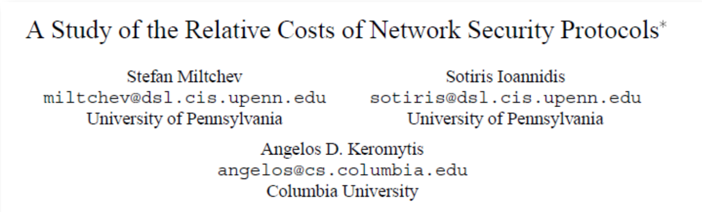


Figure 10: Energy consumption for client and server operations in SSL handshake under the presence or absence of client authentication

The Cost of Security

Cost of unprotected transfer (http) vs. secured transfer (either SSL protected transfer using https or http over IPSec)



A Study of the Relative Costs of Network Security Protocols*

Stefan Miltchev
miltchev@dsl.cis.upenn.edu
University of Pennsylvania

Sotiris Ioannidis
sotiris@dsl.cis.upenn.edu
University of Pennsylvania

Angelos D. Keromytis
angelos@cs.columbia.edu
Columbia University

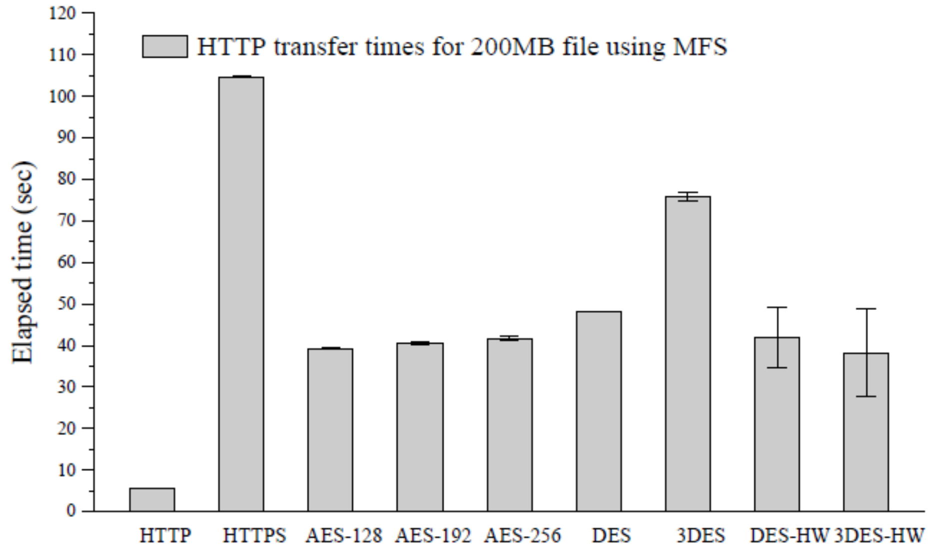


Figure 12: **Large file transfer using http, https, and http over IPsec, on a host-to-host network topology. The file is read and stored in the Unix memory file system (MFS).**

Lightweight Crypto

Performance of Lightweight Crypto

SECURE MOBILE NETWORKING

Mobile and wireless devices are on the rise, as are embedded systems with communication capabilities

- Can we use traditional crypto on resource constraint devices?
- What are the performance and energy trade-offs?

- In the following, we will give some insights into the performance of popular crypto algorithms as well as selected lightweight algorithms specially designed for constraint devices

- Source:

A Survey of Lightweight-Cryptography Implementations

Thomas Eisenbarth
Ruhr University Bochum

Christof Paar and Axel Poschmann
Ruhr University Bochum

Sandeep Kumar
Philips Research Europe

Leif Uhsadel
Catholic University of Leuven

Editor's note:
The tight cost and implementation constraints of high-volume products, including secure RFID tags and smart cards, require specialized cryptographic implementations. The authors review recent developments in this area for symmetric and asymmetric ciphers, targeting embedded hardware and software.

—Patrick Schaumont, Virginia Tech

Hardware Implementations

SEEMOO SECURE MOBILE NETWORKING

Hardware implementations of various ciphers

- Partially limited in functionality (such as encryption only)
- Collected from various sources

Table 1. Comparison of lightweight ciphers.

Cipher	Key bits	Block bits	Cycles per block	Throughput at 100 kHz (Kbps)	Logic process	Area (GEs)
Block ciphers						
Present	80	64	32	200.00	0.18 µm	1,570
AES	128	128	1,032	12.40	0.35 µm	3,400
Hight	128	64	34	188.20	0.25 µm	3,048
Clefia	128	128	36	355.56	0.09 µm	4,993
mCrypton	96	64	13	492.30	0.13 µm	2,681
DES	56	64	144	44.40	0.18 µm	2,309
DESXL	184	64	144	44.40	0.18 µm	2,168
Stream ciphers						
Trivium[5]	80	1	1	100.00	0.13 µm	2,599
Grain[5]	80	1	1	100.00	0.13 µm	1,294

*AES: Advanced Encryption Standard; DES: Data Encryption Standard; DESXL: lightweight DES with key whitening.

Software Implementations

SECURE MOBILE NETWORKING

Table 2. Comparison of software implementations of ciphers.

Cipher	Key size (bits)	Block size (bits)	Encryption (cycles/ block)	Throughput at 4 MHz (Kbps)	Decryption (cycles/ block)	Relative throughput (% of AES)	Code size (bytes)	SRAM size (bytes)	Relative code size (% of AES)
Hardware-oriented block ciphers									
DES	56	64	8,633	29.6	8,154	38.4	4,314	0	152.4
DESXL	184	64	8,531	30.4	7,961	39.4	3,192	0	112.8
Hight	128	64	2,964	80.3	2,964	104.2	5,672	0	200.4
Present	80	64	10,723	23.7	11,239	30.7	936	0	33.1
Software-oriented block ciphers									
AES	128	128	6,637	77.1	7,429	100.0	2,606	224	100.0
IDEA	128	64	2,700	94.8	15,393	123.0	596	0	21.1
TEA	128	64	6,271	40.8	6,299	53.0	1,140	0	40.3
SEA	96	96	9,654	39.7	9,654	51.5	2,132	0	75.3
Software-oriented stream ciphers									
Salsa20	128	512	18,400	111.3	NA	144.4	1,452	280	61.2
LEX	128	320	5,963	214.6	NA	287.3	1,598	304	67.2

*IDEA: International Data Encryption Algorithm; TEA: Tiny Encryption Algorithm; SEA: Scalable Encryption Algorithm.

* „All the discussed ciphers were implemented for 8-bit AVR microcontrollers. AVRs are a popular family of 8-bit RISC microcontrollers. The ATmega family offers 8 Kbytes to 128 Kbytes of flash memory and 1 Kbyte to 8 Kbytes of SRAM. The devices of the ATmega serieshave 32 general-purpose registers with a word size of 8 bits. Most of the microcontrollers' 130 instructions are one cycle, and the microcontrollers can be clocked at up to 16 MHz" (from the paper)

Wait, what about Public Key Algorithms

Results From Real Sensor Nodes

- TelosB: MSP430 microcontroller (16-bit), 48 KB ROM, 10 KB RAM
- Econotag: ARM7 microcontroller (32-bit), 80 KB ROM, 96 KB RAM

TABLE I
CODE SIZES

	TelosB	Econotag
ECDSA	1108	872
ECC	2220	1958
NN	5980	5800
SHA1	2690	1088
AES	5856	5240
HMAC-SHA1	362	280

COMPUTATION TIME

	TelosB	Econotag
ECDSA	$142s$	$14s$
DH	$38s$	$7s$
AES	$45\mu s$	$1.8ms$
HMAC-SHA1	$146\mu s$	$2.7ms$

Acks & Recommended Reading

Selected slides of this chapter courtesy of

- Radia Perlman (Intel/SUN Microsystems), Nikita Borisov (UIUC)
- Input from various research papers, notably
  - "A Study of the Relative Costs of Network Security Protocols" by Miltchev, Ioannidis, Keromytis
  - "A Survey of Lightweight-Cryptography Implementations" by Eisenbarth, Kumar, Paar, Poschmann, Uhsadel
  - "Analyzing the Energy Consumption of Security Protocols" by Potlapally, Ravi, Raghunathan, Jha

Recommended reading

- A variety of good crypto literature exists
- A freely available textbook on applied crypto is
  - "Handbook of Applied Cryptography" by Menezes, van Oorschot, Vanstone, vailable online http://www.cacr.math.uwaterloo.ca/hac/ for download, but see copyright note

Copyright Notice

Contact

Prof. Dr.-Ing. Matthias Hollick
Department of Computer Science

SEEMOO
Mornewegstr. 32
64293 Darmstadt/Germany
matthias.hollick@seemoo.tu-darmstadt.de

Phone +49 6151 16-70920
Fax +49 6151 16-70921
www.seemoo.tu-darmstadt.de