SPAKE: A Single-Party Public-Key Authenticated Key Exchange Protocol for Contact-Less Applications

Jean-Sébastien Coron[1], Aline Gouget[2,3], Pascal Paillier[2,3], and Karine Villegas[3]

[1] University of Luxembourg
[2] CryptoExperts
[3] Gemalto Security Labs

Abstract. SPAKE is a cryptographic protocol that provides lightweight transactions in contact-less applications. In this protocol a verifier (a reader or terminal) authenticates a prover (a contact-less card) relative to a certification authority. Additionally, the prover and the verifier must establish a session key for secure messaging. Contrarily to previous solutions such as MIFARE, the protocol is *asymmetric* in order to allow SAM[1]-less, low cost readers. Because contact-less transactions are subject to very strong time limitations, the protocol also achieves *high-speed computations* while providing a customizable security level.

1 Introduction

In typical contact-less transactions, authentication between a card and a reader is based on proprietary symmetric cryptography, such as MIFARE Classic [11], CALYPSO [4] or FELICA [18]. Usually, a dedicated hardware circuit is embedded in both contact-less cards and readers, and a common secret key is shared between the two parties. This architecture may suffer from dramatic security weaknesses when relying on adhoc cryptographic mechanisms, as exemplified by the recent attacks on MIFARE [12,13,14,15]. Until now, it seems that little or no industrial effort has been undertaken to design cryptographic replacements that would provide both efficiency and provable security.

We describe SPAKE (Single-party Public-key Authenticated Key Exchange), a protocol that allows fast authentication for contact-less applications. In SPAKE, a verifier called Proximity Coupling Device (PCD) authenticates a prover called Proximity Integrated Circuit Card (PICC) relative to some certification authority. Additionally, the PCD and PICC establish a session key for secure messaging and user-dependent data are then securely transmitted by the PICC to the PCD.

The originality of SPAKE is that it achieves public key authentication, thereby allowing SAM-less, low cost readers. The protocol is also fast because the card's data must be sent within very strong time limitations, namely about 150 milliseconds. The main targeted applications for this protocol are access control

[1] SAM: Secure Application Module.

R. Sion et al. (Eds.): FC 2010 Workshops, LNCS 6054, pp. 107–122, 2010.

and transport. Our SPAKE protocol is based on a public-key encryption scheme called *RSA for paranoids*, a variant of RSA designed by Adi Shamir [17] that enjoys very fast decryption. This is especially useful in our context since decryption must be performed inside the smart-card, where a cryptographic coprocessor is commonly available. The security requirements for SPAKE are the following: (i) Chip unforgeability (it should be impossible to authenticate as a PICC without knowing that PICC's private key sk); (ii) Channel secrecy (it should be infeasible to recover the session key K of a recorded transaction).

In this paper, we describe a formal security model for these two security properties. We first provide a security analysis of the SPAKE protocol generically in term of the underlying public-key encryption primitive. We show that SPAKE is secure (in the ideal cipher model) if the underlying public-key scheme is one-way under chosen-ciphertext attack *i.e.* OW-CCA-secure [3]. We then describe a variant of RSA for paranoids that achieves the OW-CCA property. This enables to make SPAKE secure against active attacks. Additionally, we show that RSA moduli with a fixed common part can be used, without degrading the overall system security. This allows to reduce transmissions, since in this case only a fraction of the modulus needs to be transmitted from the PICC to the PCD. Next, we provide a full specification of SPAKE for various levels of security, based on either DES or AES. Finally, we report benchmarks of SPAKE performances.

2 High-Level Objectives for SPAKE

2.1 Functional Requirements

A Single-Party Public-key Authenticated Key Exchange (SPAKE) protocol is a two-party cryptographic protocol played between a prover PICC and a verifier PCD. The goals of the protocol are:

1. The PCD authenticates the PICC relative to a certification authority CA;
2. The PCD and PICC establish a session key later used for secure messaging.

We note that there is no authentication of the PCD by the PICC. This implies that any attacker can fake a PCD and establish a session key with the PICC. Formally, a SPAKE protocol is a 4-uple of probabilistic algorithms:

- Keygen: given a security parameter κ, the algorithm generates a key pair (pk, sk). The public key pk gets certified by a CA and the corresponding certificate is denoted σ. Then (pk, sk, σ) is securely transmitted to the PICC.
- Challenge: the PICC sends its certified public key (pk, σ) to the PCD who checks the certificate. Optionally the PICC can send additional data denoted $data$ (for example, a commitment). Then the PCD generates a challenge

$$chal \leftarrow \mathsf{Challenge}(pk, \sigma, data)$$

and sends $chal$ to the PICC.

– Response: the PICC uses sk to generate a pair of strings

$$(res, K) \leftarrow \mathsf{Response}(sk, chal) .$$

The response res is sent to the PCD while K is kept private by the PICC, to be used as the session key.

– Verif: the PCD computes $K' \leftarrow \mathsf{Verif}(pk, \sigma, data, chal, res)$. If $K' = \perp$, then the PCD aborts the protocol. Otherwise, it uses K' as the session key.

We require that if the PICC has computed (res, K) and the PCD has obtained K', then $K' = K$. For simplicity, we assume that the public keys are certified and that the PCD always checks the certificate sent by the PICC.

The implementations of SPAKE that we consider are somewhat dictated by the quest for optimized performances. When the hardware architectures of the reader and the contact-less card both embed a coprocessor for a blockcipher such as DES or AES, it makes sense to make SPAKE rely on that primitive. In typical contact-less applications, cards are in general equipped with a cryptographic coprocessor and SPAKE may therefore take advantage of this. A cryptoprocessor is however hardly ever available on the reader side, and a desired feature of the protocol is that a simple CPU must be powerful enough to carry out all cryptographic computations on the reader.

2.2 Security Requirements

Real-life access control applications require two security properties:

1. Unforgeability: no attacker can impersonate a PICC without knowing that PICC's private key sk.
2. Channel secrecy: no attacker can recover the session key K of an eavesdropped transaction.

During an *active* attack, the adversary $\mathcal{A}$ can additionally interact with a PICC; otherwise the attack is only passive. For unforgeability under an active attack, $\mathcal{A}$ can therefore interact with a PICC before trying to pass an authentication towards the PCD. However, since there is no authentication of the PCD, an active adversary can easily fake a PCD and establish a session key with a PICC. Therefore security against active attacks cannot be achieved for the secrecy property; in this paper we only consider passive attacks for the secrecy property. Formally, we consider the following scenario between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$:

1. $\mathcal{C}$ generates a key pair (pk, sk) and sends pk to $\mathcal{A}$, along with its public-key certificate σ.
2. Using sk, $\mathcal{C}$ simulates n protocol executions between a chip and a terminal, and sends the protocol transcripts $(T_1, \ldots, T_n)$ to $\mathcal{A}$.
3. $\mathcal{A}$ can request any of the session keys used in the transcripts $(T_1, \ldots, T_n)$.
4. (Active attack). $\mathcal{A}$ can engage in up to n protocol executions with $\mathcal{C}$, who plays the role of the PICC. $\mathcal{C}$ answers using sk. A the end of each protocol execution, $\mathcal{A}$ obtains the corresponding session key.

5. (Secrecy). $\mathcal{A}$ outputs one of the session keys matching the protocol transcripts $(T_1, \ldots, T_n)$, not previously revealed in step 3.
6. (Unforgeability). $\mathcal{C}$ plays the role of the PCD and $\mathcal{A}$ plays the role of the PICC. $\mathcal{A}$ must output res such that $\mathsf{Verif}(pk, \sigma, data, chal, res) \neq \perp$.

Definition 1. *A SPAKE protocol is said to be (t, ε)-secure if no adversary running in at most t elementary steps can impersonate a PICC or recover the session key of an observed transaction with probability greater than ε.*

The security is proven using the well-known Ideal Cipher Model (ICM). Although it is better to obtain a protocol that does not rely on this assumption, the ICM is recognized as a powerful tool for obtaining both secure and efficient constructions. We note that the ICM has recently been shown to be equivalent to the Random Oracle Model (ROM) [6].

The adversary's goal differs from the adversary's goal in [2]. Namely in [2] the adversary is only asked to distinguish between a real session key and a random one; here the adversary is required to output the full session key. It is easy to see that a scheme proven secure in the ICM can be turned into a scheme secure in the ROM in the sense of [2] and vice-versa.

We want to provably achieve unforgeability and secrecy. We consider the following scenario between an attacker and a challenger. The adversary can interact at most n times with the chip. Eventually the adversary must be able to authenticate (unforgeability) or to output a session key (secrecy).

3 The SPAKE Protocol: Generic Construction

The SPAKE protocol relies on a blockcipher E as well as on a public-key encryption scheme $\mathcal{E}$. Viewing E as an ideal cipher, we further prove that our construction is secure under appropriate security assumptions on $\mathcal{E}$.

3.1 High-Level Description of SPAKE

SPAKE makes use of a blockcipher $E : \{0,1\}^\alpha \times \{0,1\}^\beta \to \{0,1\}^\beta$ where $\{0,1\}^\alpha$ is the key-space and $\{0,1\}^\beta$ is the message and ciphertext space. We require that $\alpha \leq \beta$, which applies to both DES ($\alpha = 56$, $\beta = 64$) and AES ($\alpha = \beta = 128$). We denote by $\mathcal{M}_{pk}$ the message space of $\mathcal{E}_{pk}$ and we assume that $\{0,1\}^\alpha \subset \mathcal{M}_{pk}$. We denote by $\mathcal{D}_{sk}$ the corresponding decryption algorithm. The SPAKE protocol is as follows:

- KeyGen: generate a key pair (pk, sk) for the public-key primitive. The CA issues a certificate σ on the public key pk.
- Challenge: the PICC randomly selects $k \leftarrow \{0,1\}^\alpha$ and computes $y = E_k(0)$. It then sends (pk, σ, y) to the PCD. The PCD picks a random number $r \in \{0,1\}^\alpha$ and sends $c = \mathcal{E}_{pk}(r)$ to the PICC.

- Response: the PICC recovers $r = \mathcal{D}_{sk}(c)$. If $r = \perp$ then the PICC aborts the protocol. The PICC computes $res = E_r(k)$ and $K = r \oplus k$; it then sends res to the PCD.
- Verification: the PCD verifies the certificate σ, decrypts $k = E_r^{-1}(res)$ and checks that $E_k(0) = y$. In this case it sets $K = r \oplus k$, otherwise a failure is reported.

After authentication, the PICC and PCD both use K as the session key to initiate secure messaging.

3.2 Generic Security of SPAKE

In this section we formulate the security of SPAKE in generic terms towards the underlying public-key encryption scheme $\mathcal{E}$. We start by describing the security assumptions we will make on $\mathcal{E}$ to yield a secure protocol.

Definition 2 (OW-CPA [3]). *A public-key encryption scheme $\mathcal{E}$ is said to be (t, ε)-OW-CPA if no adversary running in time t, given a random public key pk and $c = E_{pk}(m)$ where m is generated at random in the message space, can output m with probability better than ε.*

Definition 3 (OW-CCA [3]). *A public-key encryption scheme $\mathcal{E}$ is said to be (t, ε)-OW-CCA if no adversary running in time at most t, given a random public key pk and $c = E_{pk}(m)$ where m is generated at random in the message space, can output m with probability better than ε, with oracle access to a decryption oracle for any $c' \neq c$.*

The following two theorems show that SPAKE achieves the requirements of active unforgeability and passive secrecy.

Theorem 1 (Active Unforgeability). *The SPAKE protocol is (t, ε)-secure against unforgeability under active attacks, in the ideal cipher model, assuming that $\mathcal{E}$ is (t', ε')-OW-CCA, where $t = t' - \mathtt{poly}(\kappa, q, n)$ and $\varepsilon = \varepsilon' + 2^{-\beta}$, where q is the number of queries to the ideal cipher, and n is the number of protocol transcripts.*

Theorem 2 (Passive Secrecy). *The SPAKE protocol is (t, ε)-passively secure against secrecy, in the ideal cipher model, assuming that $\mathcal{E}$ is (t', ε')-OW-CPA, where $t = t' - \mathtt{poly}(\kappa, q, n)$ and $\varepsilon = n \cdot \varepsilon' + n \cdot 2^{-\alpha}$, where q is the number of queries to the ideal cipher, and n is the number of protocol transcripts.*

The proofs of Theorems 1 and 2 are provided in Appendices A and B.

4 Revisiting RSA for Paranoids

4.1 Description

The RSA for paranoids scheme (RSAP) is an asymmetric encryption scheme defined by Adi Shamir in [17]. It consists in using an unbalanced modulus $N = pq$ and in decrypting ciphertexts only modulo the smallest prime factor p. The scheme is described as follows:

- KeyGen: given the security parameter κ and a public exponent e, generate a prime p with $|p| = \kappa$ such that $\gcd(e, p-1) = 1$. Then generate a prime q such that $p \ll q$ and compute $N = p \cdot q$. Compute $d = e^{-1} \mod (p-1)$. The public key is (N, e) and the private key is (p, d). Let γ be a parameter such that $\gamma \leq \kappa - 1$.
- Encryption: given $m \in \{0,1\}^\gamma$, compute $c = m^e \bmod N$.
- Decryption: given c, compute $m = c^d \bmod p$.

This completes the description of Shamir's scheme. It is easy to see that the decryption procedure recovers the full plaintext m because we always have $0 \leq m < 2^\gamma < p$. In Shamir's scheme, decryption is very fast because it is performed only modulo p, and the size of p can be made smaller than in textbook RSA encryption where the two prime factors are balanced.

4.2 Chosen-Ciphertext Attacks against RSAP

There exists a straightforward chosen-ciphertext attack against RSAP: one generates a random $c \in \mathbb{Z}_N$ and requests its decryption to the private key holder. Given $m = c^d \mod p$, it is easy to compute $c' = m^e \mod N$ and $\gcd(c - c', N)$ then discloses p with overwhelming probability.

For our purposes, we want to design a variant of RSAP that achieves the OW-CCA property. One option is to apply the OAEP construction [1] to encrypt m as $c = (\mathsf{OAEP}(m, r))^e \mod N$ where r is a random string of appropriate size. However, the OAEP construction is proven secure only when the underlying encryption scheme is a one-way trapdoor *permutation*, as in the case of textbook RSA. Here the RSAP is *not* a permutation since the message space $\{0,1\}^\gamma$ is much smaller than the ciphertext space $\mathbb{Z}_N$.

In the following we describe a different encoding that provably achieves the OW-CCA property. Note that since we are only interested in realizing the OW-CCA property and not the (stronger) IND-CCA property, our encoding (and the corresponding security proof) is substantially simpler than OAEP. In particular it is deterministic whereas OAEP is probabilistic.

4.3 Enhancing the Security of RSA for Paranoids

Our OW-CCA-secure variant of RSAP is defined as follows:

- KeyGen: given the security parameter κ and a public exponent e, generate a prime p with $|p| = \kappa$ such that $\gcd(e, p-1) = 1$. Then generate a prime q such that $p \ll q$ and compute $N = p \cdot q$. Compute $d = e^{-1} \mod (p-1)$. The public key is (N, e) and the private key is (p, d).

  The message space is $\{0,1\}^\alpha$. Let $H : \{0,1\}^\alpha \to \{0,1\}^\ell$ be a hash function, where the output size ℓ is such that $\alpha + \ell \leq \kappa - 1$.
- Encryption: given $m \in \{0,1\}^\alpha$, compute $c = (m \| H(m))^e \bmod N$.

– Decryption: given c, compute $x = c^d \bmod p$ and parse x as $m\|h$ where $m \in \{0,1\}^\alpha$ and $h \in \{0,1\}^\ell$. If the parsing fails or if $h \neq H(m)$, return $\perp$. Otherwise return m.

We will refer to this scheme as the RSAP-H variant. The following theorem shows that RSAP-H is OW-CCA secure assuming that RSAP is partially one-way.

Definition 4 (P-OW-CPA [5]). *A public-key encryption scheme $\mathcal{E}$ is said to be (t,ε)-partially-OW-CPA (P-OW-CPA secure) if given a random public key pk and $c = \mathcal{E}_{pk}(m)$ where $m = m_1\|m_2$ is generated at random in the message space and $m_1 \in \{0,1\}^{k_1}$, no adversary running in time t can output m_1 with probability better than ε.*

Theorem 3. *RSAP-H is (t,ε)-OW-CCA secure in the random oracle model, assuming that RSAP is (t',ε')-P-OW-CPA secure with $k_1 = \alpha$ and posing $\varepsilon = q_h \cdot \varepsilon' + q_c \cdot 2^{-\ell}$ where q_h is the number of hash queries and q_c is the number of ciphertext queries.*

We refer to Appendix C for the proof of Theorem 3.

4.4 Instantiating SPAKE with RSAP-H

We define $H(m) = E_m(0) \bmod 2^\ell$ where ℓ is a parameter such that $\ell \leq \beta$. The blockcipher E must be independent from the blockcipher used in SPAKE. This can be done by pre-pending a dedicated bit in the key input. The full SPAKE protocol based on the blockcipher $E : \{0,1\}^\alpha \times \{0,1\}^\beta \to \{0,1\}^\beta$ where again $\alpha \leq \beta$ is described as follows:

– KeyGen: given the security parameter $\kappa > \alpha + \beta$ and a public exponent e, generate a prime p with $|p| = \kappa$ such that $\gcd(e, p-1) = 1$. Then generate a prime q such that $p \ll q$ and compute $N = p \cdot q$. Compute $d = e^{-1} \bmod (p-1)$. The public key is (N, e) and the private key is (p, d). The CA issues a certificate σ on the public key pk.
– Challenge: the PICC randomly selects $k \leftarrow \{0,1\}^{\alpha-1}$ and computes $y = E_{0\|k}(0)$. The tuple (pk, σ, y) is then sent to the PCD. The PCD picks a random number $r \in \{0,1\}^{\alpha-1}$, and lets h be the ℓ least significant bits of $E_{1\|r}(0)$; it then sends c to the PICC, where $c = \big(r\|h\big)^e \bmod N$.
– Response: the PICC computes $x = c^d \bmod p$ and parses x as $r\|h$ where $r \in \{0,1\}^{\alpha-1}$ and $h \in \{0,1\}^\ell$. If the parsing fails or if h is not equal to the ℓ least significant bits of $E_{1\|r}(0)$, the PICC aborts the protocol. The PICC computes $res = E_{0\|r}(k)$ and $K = r \oplus k$. Then res is sent to the PCD.
– Verification: the PCD verifies the certificate σ, decrypts $k = E^{-1}_{0\|r}(res)$ and checks that $E_{0\|k}(0) = y$. If the verification succeeds, the PCD sets $K = r \oplus k$. Otherwise PCD aborts the protocol.

After the protocol is executed, the PICC and PCD both use $K \in \{0,1\}^{\alpha-1}$ as the session key. The passive secrecy and active unforgeability properties follow from Theorems 1, 2 and 3.

5 RSA Moduli with Predetermined Bits

In this section, we show that SPAKE supports the use of RSA moduli with a fixed common part without degrading the security properties. This enables to reduce the size of transmissions, since in this case only a fraction of the modulus needs to be transmitted between the PICC and the PCD. Thus the global process execution time could be reduced.

5.1 The Key Generation Algorithm

In [17], the following RSA generation algorithm with predetermined part is proposed (here we use slightly different notations):

Generation of an RSA modulus with a predetermined part

Input: κ, n and t, and predetermined string $s = 1\|s'$ with $s' \in \{0,1\}^{n-\kappa-t-1}$

Output: a modulus N such that $s \cdot 2^{\kappa+t} \leq N < (s+1) \cdot 2^{\kappa+t}$

1. Generate a prime p in the interval $[2^{\kappa-1}, 2^{\kappa}[$.
2. Let $a \leftarrow \lfloor s \cdot 2^{\kappa+t}/p \rfloor$
3. Let $b \leftarrow \lfloor (s+1) \cdot 2^{\kappa+t}/p \rfloor$
4. Generate a random prime q in the interval $[a, b[$.
5. Return $N = p \cdot q$

The parameter t must be large enough so that there are enough primes in the interval $[a, b[$. One can take for example $t = 50$. It is argued in [17] that an RSA modulus with a predetermined part offers the same level of security than a standard RSA modulus. Namely, one can show that the distribution of q in the previous algorithm is close to the distribution of q in a standard RSA modulus, when the predetermined part s is generated at random. Therefore, any factoring algorithm against a modulus N with predetermined part would work equally well against a standard RSA modulus.

5.2 Using a Common Predetermined Part in SPAKE

In the following we show that all users can actually share the same predetermined part s, where s is initially generated at random by the Certificate Authority. The CA will only certify moduli N with this predetermined part. Formally, we consider the following adaptation of the SPAKE protocol.

Setup: Given parameters n, κ and t, the Certificate Authority generates $s = 1\|s'$ where $s' \leftarrow \{0,1\}^{n-\kappa-t-1}$ is chosen at random. The CA publishes s.

SPAKE: The rest of the SPAKE protocol is identical to the description given in Section 3.1, except that every user will generate an RSA modulus with the same predetermined part s.

The following theorem shows that if SPAKE is secure with a single user, then it remains secure if all users share the same predetermined part in their RSA modulus.

Theorem 4. *If RSAP-H-SPAKE is secure when the modulus of one user has a predetermined part $s = 1\|s'$ where $s' \leftarrow \{0,1\}^{n-\kappa-t-1}$, then RSAP-H-SPAKE remains secure when all the moduli of users share the same predetermined part.*

Proof. We consider an attacker $\mathcal{A}$ which can adaptively corrupt all users except one. Let n be the number of users. We are given as input a public key corresponding to a single user $\mathcal{U}$, with predetermined part s. We select a random index j in $[1, n]$. Then for user j we use the public key of $\mathcal{U}$; for the other users we generate a RSA modulus with the same predetermined part s. The attacker $\mathcal{A}$ cannot distinguish between user j and the other users, because all RSA moduli follow the same distribution. With probability at least $1/n$, his attack applies against user j; in this case, this gives an attack against the original user $\mathcal{U}$. □

6 Real-Life Implementations of SPAKE

Since it is common for embedded hardware platforms to feature a coprocessor for DES or AES, we consider implementations of SPAKE using one or the other blockcipher. We consider four possible levels of security: 55 bits, 64 bits, 80 bits and 100 bits. For each of these levels of security, we specify which blockcipher to use, the required bit-size of N, the bit-size of p, the value of e and the number of predetermined bits of N.

6.1 Basing SPAKE on DES

As described in previous sections, we can use DES to instantiate the blockcipher in the SPAKE protocol, thus posing $\alpha = 56$ and $\beta = 64$. In this case, the security level is at most $\alpha - 1 = 55$ bits. To obtain a higher security level, one could think of using 3-DES instead of DES; we stress that this is not possible in the context of SPAKE. Namely the security proofs of Theorems 1 and 2 rely on the Ideal Cipher Model, and it is easy to see that 3-DES does *not* behave as an ideal cipher. To justify this claim, consider the 3-DES blockcipher

$$\mathsf{3DES}(k_1\|k_2\|k_3, m) = \mathsf{DES}(k_3, \mathsf{DES}^{-1}(k_2, \mathsf{DES}(k_1, m))) \ .$$

Assume that k_1 and k_2 are unknown to the attacker, but k_3 is known. Then given $c = \mathsf{3DES}(k_1\|k_2\|k_3, m)$, the attacker can easily compute $c' = \mathsf{3DES}(k_1\|k_2\|k'_3, m)$ for any k'_3 since $c' = \mathsf{DES}(k'_3, \mathsf{DES}^{-1}(k_3, c))$. It is easily seen that this is impossible to do with an ideal cipher. Therefore 3-DES cannot be viewed as an ideal cipher, even when DES is viewed as an ideal cipher. This implies that we cannot hope to increase the security of SPAKE by using 3-DES instead of DES.

6.2 Basing SPAKE on AES

We can also use AES-128 to instantiate the blockcipher of SPAKE, in which case $\alpha = \beta = 128$. The security level is then at most $\alpha - 1 = 127$ bits.

6.3 Tuning the Size of N and p

The RSA for paranoid scheme RSAP uses an unbalanced RSA modulus $N = pq$ with $p \ll q$, and in our search for best performance the sizes of N and p would tend to be as small as possible. The security strength [7] of generated moduli therefore depends on their resistance to integer factoring algorithms. There are two categories of factoring techniques:

1. Factoring algorithms whose running time depends on the size of N; the fastest such algorithm is the General Number Field Sieve (GNFS) [9].
2. Factoring algorithms whose running time depends on the size of p: the fastest such algorithm is the Elliptic Curve Method (ECM) [8].

For GNFS, we use the same complexity estimates as in [7], where the following expression for the security strength of an n-bit RSA modulus N is given as $s_{\mathsf{GNFS}}(n) = \left(\frac{64}{9}\right)^{1/3} \cdot \log_2(e) \cdot (n \ln 2)^{1/3} \cdot (\ln(n \ln 2))^{2/3} - 14$. For ECM, we use the following formula (see Appendix D), which gives the security strength of a κ-bit prime p: $s_{\mathsf{ECM}}(\kappa) = (\ln 2)^{-1/2} \cdot (2 \cdot \kappa \ln(\kappa \ln 2))^{1/2} + 5$. We summarize the required key size for the 55-bit, 64-bit, 80-bit and 100-bit security levels in Section 6.6.

6.4 Coppersmith's Attack and Shamir's Bound

In this section we describe an attack based on Coppersmith's theorem for finding small roots of polynomial equations. The attack applies when a small public exponent e is used.

Theorem 5 (Coppersmith). *Let $N = pq$ be a RSA modulus of unknown factorization and $f(x)$ by a polynomial of degree δ. There exists a polynomial-time algorithm that finds all roots x_0 of $f(x) = 0 \mod N$ such that $0 \leq x_0 < N^{1/\delta}$.*

If $m^e < N$, then the message can be recovered by taking an e-th root in $\mathbb{Z}$. Coppersmith's algorithm allows to go beyond this bound using exhaustive search. More specifically, we consider the RSA for paranoids encryption scheme $c = m^e \bmod N$ where $m \in \{0,1\}^\gamma$. We write $m = m_0 \cdot \lfloor N^{1/e} \rfloor + x_0$ where $x_0 < N^{1/e}$. This gives $c = (m_0 \cdot \lfloor N^{1/e} \rfloor + x_0)^e \mod N$. If we are given m_0, this gives a polynomial equation of degree e in x which has a small root $|x_0| < N^{1/e}$. We can then apply Coppersmith's theorem and recover the full value of m. Since m_0 is unknown, we must perform an exhaustive search on m_0 and apply Coppersmith's theorem for each possible value of m_0. Since the size of m is γ bits, the size of m_0 in bits is $|m_0| = \gamma - |N|/e$. For a targeted security level of k bits, we must have $|m_0| > k$, which gives the condition $\gamma \geq (\log_2 N)/e + k$, thus leading to the condition $e \geq \frac{\log_2 N}{\gamma - k}$ where k is the security parameter.

In our OW-CCA-secure variant of RSA for paranoids implemented with the blockcipher $E : \{0,1\}^\alpha \times \{0,1\}^\beta \to \{0,1\}^\beta$, we have $\gamma = \alpha + \ell - 1$. This gives the condition $e \geq \frac{\log_2 N}{\alpha + \ell - 1 - k}$. We also consider Shamir's bound in [17]. It consists in taking e such that the size of m^e before modular reduction is at least twice

the size of m, as in Rabin encryption. This gives the bound $e \geq \frac{2\log_2 N}{\alpha+\ell-1}$. In this paper, we use the strictest of those two bounds. We note that if $\alpha + \ell \geq 2k$, Coppersmith's bound is automatically satisfied when Shamir's bound is satisfied. This is the case for the parameters considered in this paper.

6.5 Predetermined Bits

We refer to the algorithm of Section 5. Given an n-bit modulus with $|p| = \kappa$ and $p \ll q$, the size of the predetermined part is set to $n - \kappa - t - 1$ bits, where t is a parameter that must be large enough so that the prime generation algorithm succeeds with overwhelming probability. One can take for example $t = 50$. Then the size of the predetermined part is $n - \kappa - 51$ bits. The size of the remaining bits which are different for each user is therefore $\lambda = \kappa + 51$ bits.

6.6 Summarizing

In light of the previous sections, we summarize in Table 1 the parameters corresponding to the various security levels. We recommend to consider a security level of at least 80 bits, since 55 bits or 64 bits of security might not be enough in practice for secure applications.

Table 1. Security level, size of N, size of p, number λ of non-predetermined bits in N, minimal value for e, blockcipher, and key-size α and block-size β of the block cipher, and output size ℓ of the redundancy used in SPAKE with RSAP-H

SPAKE Security	$\|N\|$	$\|p\|$	λ	e	blockcipher	α	β	ℓ
55 bits	640	192	243	11	DES	56	64	64
64 bits	832	240	291	11	AES-128	128	128	104
80 bits	1248	352	403	11	AES-128	128	128	128
100 bits	2048	560	611	17	AES-128	128	128	128

7 Proof of Concept/Prototype

We have realized a proof of concept based on NXP's SmartMX P5CT072 platform, which features the FameXE cryptoprocessor and a hardware DES processor. In contactless applications the CPU clock can be set to 31 MHz, the hardware DES can be clocked at 36 MHz and the FameXE at 48 MHz. We have used a specific Mini OS for test and benchmarking purposes. The PCD was simulated on a PC via a transparent contact-less reader.

The code size of our SPAKE library is 1.6 KB. The library supports all cryptographic operations, thus excluding APDU treatment executed by the OS. Tables 2 and 3 provide the benchmarks of the various stages of a typical transport transaction in two different settings. The initialisation stage (Init) initiates the anti-collision protocol, selects the application and performs a PPS negotiation of the transmission baudrate. The *Get Challenge* APDU command consists in

running the Challenge algorithm of SPAKE, where the PICC sends the commitment y and the certificate σ. We assume that, since a significant fraction of the bits of the public key N are fixed, the signature scheme of the certificate is a Rabin-Williams signature with message recovery. Therefore only the certificate needs to be sent, and the PCD recovers N when verifying it. The *Get Response* command consists of the decryption stage using our variant of RSA for paranoids. Execution times on the PICC side are separated from the time required by contact-less transmissions, which we provide at 106 and 424 Kbits per second respectively. Also note that the code is executed from EEPROM, which is more time consuming than when the code is executed from ROM — so performances can be improved in an actual product. Finally when the *Get Data* command is played, the card sends three typical files (user data and profile w.r.t the transport application) over the secure channel to the reader.

Table 2 reports performances for $\kappa = |p| = 224$, $|N| = 512$ and $|\sigma| = 1280$. The RAM consumption in this case is about 800 bytes and 216 bytes of non-volatile memory (EEPROM) are required to store (σ, p, d). The total transaction time in this case is close to 96 milliseconds.

Table 2. Benchmarks (in microseconds) of a contact-less transaction for $\kappa = 224$

$p = 224$	Proc PICC	Com@106	Com@424
Init	1521	5400	3986
Get Challenge	7446	15873	4022
Get Response	24367	7693	1961
Get Data	7045	12120	3143
Deselect	146	727	219
Total	40525	41813	13331

Table 3 provides performances for $\kappa = 352$, $|N| = 1248$, where $|\sigma|$ is still 1280. In this case, about 900 bytes of RAM and 248 bytes of EEPROM are required. A full transaction is then completed in about 156 ms.

Table 3. Benchmarks (in microseconds) of a contact-less transaction for $\kappa = 352$

$p = 352$	Proc PICC	Com@106	Com@424
Get Response	73101	17015	4291
Total	89259	51135	15661

Countermeasures against Radio-Frequency Analysis have been undertaken. Classical protections against the RF versions of SPA/DPA and related attacks can be carried out by randomizing the private exponent d and the prime factor p. We assume that the hardware blockcipher is inherently immune against side-channel attacks.

Finally, we stress that this prototype is intended for test purposes only and that the chosen parameters provide a low security level of 49 bits in the case of Table 2. These parameters should be adjusted as indicated in Table 1 to reach an appropriate security level in actual products.

8 Conclusion

In this paper, we have provided a new protocol for authenticated key exchange. We have shown that our protocol is secure against active attacks if the underlying public-key scheme is One-Way Chosen-Ciphertext (OW-CCA) secure. For this we have designed a variant of RSA for paranoids that achieves that OW-CCA property. Additionally, we have shown that RSA moduli with a fixed common part can be used, without degrading the overall system security. This enables to reduce the communication bandwidth, since in this case only a fraction of the modulus needs to be transmitted between the PICC and the PCD, when the certificate does not support message recovery. Then, we have provided a full specification of SPAKE for various levels of security. Finally, the details of a prototype have been reported along with performance benchmarks.

References

1. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
2. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, p. 139. Springer, Heidelberg (2000)
3. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations Among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, p. 26. Springer, Heidelberg (1998)
4. Technical documents available, `http://www.calypsotechnology.net/`
5. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is Secure under the RSA Assumption. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 260. Springer, Heidelberg (2001)
6. Coron, J.S., Patarin, J., Seurin, Y.: The Random Oracle Model and the Ideal Cipher Model are Equivalent. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 1–20. Springer, Heidelberg (2008)
7. European Network of Excellence ECRYPT, Yearly Report on Algorithms and Keysizes (2007-2008), `www.ecrypt.eu.org/ecrypt1/documents/D.SPA.28-1.1.pdf`
8. Lenstra Jr., H.W.: Factoring Integers with Elliptic Curves. Ann. Math. 126, 649–673 (1987)
9. Lenstra, A.K., Lenstra Jr., H.W.: The development of the number field sieve. Lecture Notes in Math, vol. 1554. Springer, Heidelberg (1993)
10. Girault, M., Poupard, G., Stern, J.: On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. Journal of Cryptology 19(4), 463–487 (2006)
11. NXP Semiconductors. MF1ICS70 functional specification (January 2008), `http://mifare.net`

12. Nohl, K., Plötz, H.: Little Security, Despite Obscurity. In: Chaos Communication Congress
13. Nohl, K.: Mifare security. In: Chaos Communication Congress
14. Courtois, N., Nohl, K., O'Neil, S.: Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards. Cryptology ePrint Archive, http://eprint.iacr.org/2008/166
15. Courtois, N.: Conditional Multiple Differential Attack on MiFare Classic. In: Rump session of Eurocrypt 2009 (2009)
16. SAGE mathematics library, http://www.sagemath.org
17. Shamir, A.: RSA for paranoids. CryptoBytes 1, 1–4 (1995)
18. Sony Global - FeliCa Web Site, Technical documents available, http://www.sony.net/Products/felica/
19. Zimmermann, P.: The ECMNET Project, http://www.loria.fr/~zimmerma/records/ecmnet.html
20. Zimmermann, P., Dodson, B.: 20 Years of ECM. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 525–542. Springer, Heidelberg (2006)

A Proof of Theorem 1

We receive a public key pk and a challenge $c^* = \mathcal{E}_{pk}(m^*)$, and we must recover m^*, using an adversary that breaks the active unforgeability property with probability ε. We also have access to a decryption oracle that can decrypt any ciphertext except c^*.

The ideal cipher E is simulated in the standard way. We start our interaction with the PICC by generating the n protocol transcripts; we can do this without knowing the private key sk. Eventually the adversary tries to impersonate the PICC. In the active attack phase, we use the decryption oracle from the OW-CCA challenge in order to decrypt the ciphertext sent by the adversary; for the rest we proceed as in the passive attack phase.

Eventually the adversary tries to impersonate the PICC. The adversary first sends the commitment y^*; we send the challenge ciphertext c^* to the adversary. The adversary sends res.

To solve the OW-CCA challenge we look at the history of E; for all triples of the form (r, k, res), we compute $c = \mathcal{E}_{pk}(r)$. If $c = c^*$ we return r as the decryption of c^*.

We now provide an analysis of the success probability of our reduction. Our simulation of E is perfect; moreover, our ciphertext c^* has the same distribution has in the original attack scenario. Therefore, when interacting with our simulation the attacker succeeds with the same probability as in the original attack scenario, that is with probability at least ε; in this case by definition we must have $y = E_k(0)$ where $k = E^{-1}_{m^*}(res)$.

If there is no triple of the form $(m^*, \cdot, res)$ in the history of E, then the distribution of $k = E^{-1}_{m^*}(res)$ is independent from the adversary's view; then the probability that $y = E_k(0)$ is at most $2^{-\beta}$. Therefore, with probability at least $\varepsilon - 2^{-\beta}$, the triple (m^*, k, res) belongs to history of E. Since by assumption the PK encryption scheme is deterministic, our reduction can check that c^* is indeed the encryption of m^* and therefore output m^*. Therefore, with probability at

least $\varepsilon - 2^{-\beta}$, our reduction outputs the correct solution m^* to the OW-CCA challenge.

B Proof of Theorem 2

We receive a public key pk and a challenge $c^* = \mathcal{E}_{pk}(m^*)$, and we must output m^*. The ideal cipher E is simulated in the standard way. We also generate the protocol transcripts as in the proof of Theorem 1, except that for a randomly chosen index j in $[1, n]$, we use the challenge ciphertext c^* instead of $c = \mathcal{E}_{pk}(r)$. More precisely, for the j-th protocol transcript, we generate a random $k \in \{0,1\}^\alpha$ and let $y = E_k(0)$; we also generate a random $res \in \{0,1\}^\beta$; the corresponding j-th transcript is then (y, c^*, res); this implicitly defines $E_{m^*}(k) = res$. If the adversary requests the session key for this j-th transcript, we abort. Eventually, the attacker outputs the session key corresponding to one of the previous n transcripts, not previously revealed.

To solve the OW-CPA challenge, our reduction determines the list of triples of the form $(r, \cdot, res)$ in the history of E, where res is the response in the j-th transcript, and for each of these triples it determines whether $c = \mathcal{E}_{pk}(r)$; in this case, it outputs r as a solution to the OW-CPA challenge.

Now we analyze the success probability of our reduction. We denote by S the event that in the original attack scenario, the adversary eventually outputs the session key for the j-th transcript, for an index j chosen uniformly at random in $[1, n]$; we have: $\Pr[\mathsf{S}] \geq \frac{\varepsilon}{n}$. We denote by Bad the event that the adversary makes a query for $E_{m^*}^{-1}(res)$. We have that conditioned on $\neg\mathsf{Bad}$, our simulation of E is perfect and the adversary's view has the same distribution as in the original scenario; therefore, the event Bad has the same probability in the original scenario and in our reduction.

Moreover, if event Bad does not occur in the original scenario, then the adversary's view is independent from the value of k in the j-th transcript; therefore, the probability that the adversary outputs $K = r \oplus k$ is at most $2^{-\alpha}$, which gives: $\Pr[\mathsf{S}|\neg\mathsf{Bad}] \leq 2^{-\alpha}$. We have $\Pr[\mathsf{S}] = \Pr[\mathsf{S}|\mathsf{Bad}] \cdot \Pr[\mathsf{Bad}] + \Pr[\mathsf{S}|\neg\mathsf{Bad}] \cdot \Pr[\neg\mathsf{Bad}] \leq \Pr[\mathsf{Bad}] + \Pr[\mathsf{S}|\neg\mathsf{Bad}]$. Therefore, we obtain that $\Pr[\mathsf{Bad}] \geq \Pr[\mathsf{S}] - \Pr[\mathsf{S}|\neg\mathsf{Bad}] \geq \frac{\varepsilon}{n} - 2^{-\alpha}$. Finally, we have that our reduction succeeds if event Bad occurs. Namely since by assumption the underlying PK encryption scheme, we can check that c^* is indeed the encryption of m^* and therefore output m^*. This gives that $\Pr[\mathsf{Succ}] \geq \frac{\varepsilon}{n} - 2^{-\alpha}$.

C Proof of Theorem 3

We receive a P-OW-CPA challenge (N, e, y^*) where $y^* = (x_1 \| x_2)^e \mod N$, where $x_1 \in \{0,1\}^\alpha$ and $x_1 \| x_2$ is randomly generated in $\{0,1\}^\gamma$, where $\gamma = \alpha + \ell$; our goal is to recover x_1. The adversary is run with public key (N, e) and target ciphertext y^*; this implicitly defines $H(x_1) = x_2$. We answer H-queries and ciphertext queries as follows:

H-queries: given a fresh hash query for $H(m)$, we generate a random $h \in \{0,1\}^\ell$ and store (m, h) in a H-table; then we return h.

Ciphertext queries: given a ciphertext query c, we proceed as follows. For all (m, h) in H-table: compute $c' = (m\|h)^e \mod N$; if $c = c' \mod N$, return m; if $\gcd(c - c', N) = p$, return m. If $\gcd(c, y^*) = p$, recover d and x_1, and return x_1. Else return $\perp$.

Adversary "type 1": the adversary does not make a hash query to x_1. The simulation of H is perfect. Consider a ciphertext query for c, and let $x = c^d \mod p$. We denote by $\mathcal{D}$ the regular decryption oracle and $\mathcal{S}_D$ our simulated oracle. Then, if x cannot be parsed as $m\|u$, then both $\mathcal{D}$ and $\mathcal{S}_D$ return $\perp$. Else, x can be parsed as $m\|u$ and there are two cases. If (m, u) is in H-table, then both $\mathcal{D}$ and $\mathcal{S}_D$ return m. Else, (m, u) is not in H-table, and:

1. If $m = x_1$, then if $u = x_2$ both $\mathcal{D}$ and $\mathcal{S}_D$ return x_1, otherwise both return $\perp$.
2. If $m \neq x_1$, then $\mathcal{D}$ returns m if $u = H(m)$ and $\perp$ otherwise, whereas $\mathcal{S}_D$ always returns $\perp$.

Therefore, $\mathcal{D}$ and $\mathcal{S}_D$ only differ for case 2, which happens with probability at most $2^{-\ell}$. Since there are at most q_c ciphertext queries, our simulation of $\mathcal{D}$ is perfect except with probability at most $q_c \cdot 2^{-\ell}$. Therefore the adversary eventually outputs x_1 with probability at least $\varepsilon' \geq \varepsilon - q_c \cdot 2^{-\ell}$.

Adversary "type 2": the adversary makes a hash query for x_1. In this case, our simulation selects a random query among the list of H queries; therefore our simulation outputs x_1 with probability at least $\varepsilon' \geq \frac{\varepsilon - q_c \cdot 2^{-\ell}}{q_h}$, where q_h is the number of hash queries.

D Security against the ECM Algorithm

The largest prime factor found using the ECM is a 222-bit integer (a table of the largest factors found by the ECM is maintained in [19]). It is estimated in [20] that the factorization of a 216 bits prime factor takes 24 years on a single 2.4 GHz PC, which corresponds to 2^{61} operations. Moreover the complexity of the ECM is $C(p) = \exp\left((\sqrt{2} + \circ(1))\sqrt{\log p \log\log p}\right)$. Ignoring the $\circ(1)$ term and using $C'(p) = C_0 \cdot \exp\left((\sqrt{2} + \circ(1))\sqrt{\log p \log\log p}\right)$ instead (where C_0 is a constant), the security level in bits can be estimated as $\log_2 C'(p) = \log_2 C_0 + \frac{\sqrt{2} \cdot \log p \log\log p}{\log 2}$ where C_0 is taken such that $\log_2 C'(2^{216}) = 61$. We obtain the following formula, which gives the security level of a κ-bit prime p: $s_{\mathsf{ECM}}(\kappa) = (\ln 2)^{-1/2} \cdot (2 \cdot \kappa \ln(\kappa \ln 2))^{1/2} + 5$.