Efficient Authenticated Key-Exchange for Devices with a Trusted Manager[*]

He Ge Stephen R. Tate
Department of Computer Science and Engineering
University of North Texas
Denton, TX 76203

Abstract

We propose an efficient authentication method for secure communication among a set of devices that have a single trusted manager or administrator, with protocols presented for authentication and authenticated key exchange. An example of such a setting would be a set of devices or sensors owned by a single person, including applications such as a smart house or coordinated control systems, with emphasis on the simplicity and efficiency of the protocol. While known techniques can solve this problem, we show how specific properties of our setting can allow our more efficient solution, which is more appropriate for embedded processors with limited computational capabilities. Specifically, a device using our protocol can authenticate itself using only about 15% of the computation required by a standard RSA signature-based authentication. We prove that our scheme is secure under the strong RSA assumption and the computational Diffie-Hellman assumption.

Keywords: Authentication, Key Exchange, Strong RSA Assumption, Embedded Systems

1. Introduction

Consider the following scenario: A person owns several devices which are capable of communication and interaction, but these devices use embedded processors whose computational capabilities are limited as compared to desktop computers. Examples of this scenario include entertainment devices or appliances owned by a consumer, multiple control and sensor systems in an automobile or airplane, and environmental controls in a building. We would like these devices to use encrypted communication for confidentiality and integrity, and to be able to securely recognize other devices with the same owner. A pair of devices should be able to establish authenticated secure channels on their own, without needing to interact with the owner or any other party at the time, and we want the system to be dynamic, meaning that new devices can enter the system at any time. Therefore, solutions based on symmetric cryptography, whether requiring preloaded keys for all pairs of devices or using an always-available authority (like in Kerberos), do not satisfy our requirements.

There are many known cryptographic solutions which can solve this problem, including using X.509 certificates issued by the owner, using a group signature or credential system with the owner as the group manager, or using identity-based encryption. However, our scenario has some unique properties which allow for a more efficient solution, and the importance of highly optimized protocols in embedded systems has led us to develop such an efficient solution, which we present in this paper. After defining a model capturing the specific characteristics of our setting in the next section, we will discuss how various aspects of our model allow us to improve upon these existing techniques.

2. Model and Relation to Previous Work

In this section, we define a model which captures security requirements in a system with three key properties: a small set of devices (typically a few devices up through a few thousand sensors), a single completely trusted administrator (the owner of the devices), and no externally-meaningful names that need to be linked to keys. To protect communication between members of this set of devices, it is necessary to provide a security protocol to implement authentication and key establishment. In the following definition we precisely specify the security requirements of our system. We use terminology from the related area of group signatures, but we caution the reader to keep in mind an important difference in our setting: the group members are not independent people, but rather the devices ultimately controlled by the single owner. We also clarify here that any attacker is assumed to be a probabilistic polynomial time algorithm, so references to things being impossible (such as forging a key) should be understood to mean impossible for a probabilistic polynomial time algorithm.

[*]This research is supported in part by NSF award 0208640.

Definition 2.1 (The Model) *A group consists a single administrator, and at most several thousand group members. The administrator holds a* group master key *while each group member holds its* group member key. *A system supporting authenticated key exchange should satisfy following properties:*

1. *(Forgery-resistance) A valid group member key can only be produced with knowledge of the group master key, so if only the administrator has access to this key then only the administrator can create valid group member keys.*
2. *(Authentication) A party can prove membership in the group if and only if it knows a valid group member key.*
3. *(Key Exchange) Any two members in the group can compute a shared secret that cannot be deduced by outsiders. This shared secret can be used with symmetric cryptography to support confidentiality and integrity of subsequent communication.*
4. *(Robustness) The compromise of a group member will not affect the security of interactions between uncompromised group members.*[1]

In addition to the above required properties, systems may also support the following two optional properties.

5. *(Forward secrecy) If the administrator's group master key is obtained, an attacker still can not compromise operations between group members whose keys were established prior to the administrator compromise.*[2]
6. *(Member Revocation) The administrator can revoke a group member in case it does not belong to the group anymore, or it is broken by an attacker.*

The importance of the last two properties depends on the likelihood of compromise of the administrator or the group members, respectively. If the group master key is kept on a general-purpose computer connected to the Internet, it is more important to consider forward secrecy than in an environment in which the group master key is kept on a dedicated device with highly controlled access. Similarly, in settings such as an unattended sensor network in a hostile environment, it is almost inevitable that group members will be compromised, so revocation is vital. In other settings, such as fixed devices in a physically controlled environment (like appliances in a home) the chances of member compromise is low, so revocation is less important.

2.1. Our Result and Related Work

In this paper we propose a simple and efficient authentication scheme which can be deployed in applications which satisfy the above model. Its security is based on the strong RSA assumption and the computational Diffie-Hellman assumption. In this section we show how our scheme is related to other solutions for this problem, and describe which properties of our system enable us to improve upon these other solutions.

X.509 certificates [12] provide an obvious solution for authentication, which is what SSL does for connections over the Internet. While this does provide the group member authentication that we need, the primary purpose of X.509 certificates is to bind a meaningful identity and other properties to a cryptographic key, which is unnecessary in our setting. To support the flexible requirements of X.509 certificates, a typical certificate would be a separate object that is several times as large as the key that it is authenticating. For example, the current X.509 certificate authenticating the 128 byte (i.e., 1024 bit) key for `www.amazon.com` is 945 bytes long, so requires 738% space overhead. By contrast, our authentication is implicit in the key itself, so requires no overhead at all (although if forward secrecy is needed we must double the size of the key). Furthermore, a device which wants to authenticate itself using an X.509 certificate must perform an RSA signature, using a private exponent that is as long as the modulus, so an RSA signature with a 1024 bit key would require an average of 1536 modular multiplications. By contrast, our scheme uses a short (160-bit) private exponent, allowing a device with a 1024 bit key to authenticate itself with an average of 240 modular multiplications — roughly 15% of the number required by the RSA signature. Note that this improvement assumes that the group manager is not compromised. If group manager compromise is a possibility, then to attain the forward-secrecy property we need double the size of the public modulus n, which would add more computing cost to the system.

Identity-based encryption (IBE) [14, 6] can also be used to solve our basic problem, but these schemes are somewhat less efficient than our system. More important than efficiency in this case is the inability of IBE schemes to support forward secrecy of keys. Since the basis of these schemes is the ability for the key manager to compute a private key from the corresponding public key, if the group manager is compromised then all group member keys are immediately compromised.

Self-certified public keys [11, 13], *certificate-less public keys* [1], and *group signature schemes* [2] also provide the capabilities that we need. However, the basic model for these systems includes independent parties as the authenticated entities, and they do not trust the group manager. In our case, the group members are passive devices that are

[1] This is in contrast to the shared-key scheme in which the compromise of one group member would reveal all the communication in the system.

[2] Note that our forward secrecy property is a property of authenticated keys, not messages. The more basic property of forward secrecy of messages applies to encryption systems, not key exchange, but we point out that since our key exchange establishes random session keys any larger system using our key exchange protocol will also exhibit forward secrecy of messages.

owned by, and hence completely trust, the group manager. The effort that these techniques use to protect against a malicious group manager is unnecessary in our case, since everyone is "on the same team." In addition, properties such as anonymity and unlinkability, as supported by group signatures, have no meaning in our system, so are unnecessary complications.

Our system has a few additional properties which make it especially appropriate for computation and memory constrained devices. First, as mentioned above, we use short private exponents, greatly improving the complexity of authentication. While many parts of our system map to standard RSA, what would be the public key in RSA is hidden in our system (our public key is g^e rather than e, so extracting e would be solving the discrete log problem), so attacks on RSA with a small decryption exponent [5] do not work against our system. Second, keys are bound to short, unique tags, that are typically 16 or 24 bits. Hence, if revocation needs to be supported, it can be done through a very compact revocation list. In addition, we provide an efficient secure re-keying scheme for use in highly dynamic environments.

3. Preliminaries

In this section, we review some definitions and widely accepted complexity assumptions that we will use in this paper [2, 8, 7].

Definition 3.1 ***(Special RSA Modulus)*** *An RSA modulus $n = pq$ is called special if $p = 2p' + 1$ and $q = 2q' + 1$ where p, q, p' and q' are all prime numbers.*

Definition 3.2 ***(Quadratic Residue Group QR_n)*** *Let Z_n^* be the multiplicative group modulo n, which contains all positive integers less than n and relatively prime to n. An element $x \in Z_n^*$ is called a* quadratic residue *if there exists an $a \in Z_n^*$ such that $a^2 \equiv x \pmod n$. The set of all quadratic residues of Z_n^* forms a cyclic subgroup of Z_n^*, which we denote by QR_n. If n is the product of two distinct primes, then $|QR_n| = \frac{1}{4}|Z_n^*|$.*

The following two properties are useful for our proofs, and have straightforward proofs based on standard number theoretic properties and properties of cyclic groups.

Property 1 *If n is a special RSA modulus, with p, q, p', and q' as in Definition 3.1 above, then $|QR_n| = p'q'$ and $(p' - 1)(q' - 1)$ elements of QR_n are generators of QR_n.*

Property 2 *If g is a generator of QR_n, then $g^a \pmod n$ is a generator of QR_n if and only if GCD$(a, |QR_n|) = 1$.*

The security of our techniques relies on the following two assumptions, which are widely accepted in the cryptography literature (see, for example, [3, 4]).

Assumption 1 ***(Strong RSA Assumption)*** *Let n be a special RSA modulus. The* Flexible RSA Problem *is the problem of taking a random element $u \in Z_n^*$ and finding a pair (v, e) such that $e > 1$ and $v^e \equiv u \pmod n$. The* Strong RSA Assumption *says that no probabilistic polynomial time algorithm can solve the flexible RSA problem with non-negligible probability.*

Assumption 2 ***(Computational Diffie-Hellman Assumption for QR_n)*** *Let n be a special RSA modulus, and let g be a generator of QR_n. Then given random g^x and g^y, it is hard to compute $g^{xy} \pmod n$.*

We also introduce a slight variant of the Strong RSA Assumption, in which we don't need to find the exponent e itself, but rather find a hidden version of it, g^e. Note that extracting the actual exponent from g^e would require solving the discrete logarithm problem over QR_n.

Assumption 3 ***(Hidden Exponent Strong RSA Assumption)*** *Let n be a special RSA modulus, and g be a generator of QR_n. Let $\log_g x$ denote the base g discrete log of x over QR_n, so $g^{\log_g x} \equiv x \pmod n$ for all $x \in QR_n$. The* Hidden Exponent Flexible RSA Problem *is the problem of taking a random element $u \in Z_n^*$ and finding a pair (v, w) such that $w \neq g$ and $v^{\log_g w} \equiv u \pmod n$. The* Hidden Exponent Strong RSA Assumption *says that no probabilistic polynomial time algorithm can solve the hidden exponent flexible RSA problem with non-negligible probability.*

We note that there is a one-to-one mapping between solutions to the flexible RSA problem and the hidden exponent flexible RSA problem. Among other things, this means that if our variant with a particular fixed hidden exponent can be solved efficiently, then the corresponding "non-hidden" exponent is a weak encryption exponent for RSA. While such exponents do exist (for example, corresponding to small decryption exponents), there is no algorithm that we're aware of that can produce a weak encryption exponent without knowledge of the factorization of n (or, equivalently, the decryption exponent). Note that if we could find such weak encryption exponents efficiently, then we could solve the standard flexible RSA problem, so it is unlikely that an attacker could find such a weak exponent to solve the hidden exponent flexible RSA problem.

4. The Proposed Scheme

The group administrator sets various parameters, the lengths of which depend on a *security parameter*, which we denote by σ, which is the length in bits of the prime factors of our public modulus n. Additional length parameters are constrained by σ, and are defined as follows:

- l_s: l_s is the bit-length of a group member private key, with the restriction that $l_s < \sigma - 1$ so that all member private keys are guaranteed to be less than $\min(p', q')$. l_s also needs to be large enough where a brute force search is not feasible.

- l_t: l_t is the bit-length of the group member public key identifiers/tags, with the restriction that $l_t < l_s$. The only lower bound on this parameter is that it must be large enough so that every group member can be assigned a distinct prime number of length l_t. Some typical examples for this length might be 8 (for groups of up to 23 members), 16 (for groups of up to 3030 members), 24 (for groups of up to 513,708 members), or 32 (for groups of up to 98,182,656 members).

Based on the speed of current number theoretic algorithms and cryptanalytic attacks on current hardware, we suggest that if forward secrecy is not needed then for moderate sized groups of up to a few thousand members, settings of $\sigma = 512$ (so n is 1024 bits), $l_s = 160$, and $l_t = 24$ offer strong security. If forward secrecy is required, then it's necessary to double σ to 1024, but the other parameters remain the same.

4.1. Key Generation

The group master key generated by the administrator is simply a special RSA modulus n as defined in Definition 3.1 where p and q are each at least σ bits long (so $p, q > 2^\sigma$), along with a generator g of the cyclic group QR_n. n and g are public values while p and q are kept secret by the administrator.

The method for the creation of a group member key is straightforward. The administrator picks two random prime numbers s and t with lengths l_s and l_t, respectively, where t has not been previously used for a different group member. The administrator computes

$$E = g^{s^{-1}t^{-1}} \pmod{n},$$

where s^{-1}, t^{-1} are the inverses of s, t modulo $|QR_n| = p'q'$, respectively. s is the private part of a group member key, while (E, t) is the public part. t is the unique identifier, or tag, to represent a group member key.

After the administrator securely delivers the group key to a group member, it stores (E, t) in its database and destroys its copy of the private key s. Therefore, even if the administrator is compromised, the private key for each group member remains secure to certain degree. We will explain this further in Section 5.

4.2. Authentication Protocol

Suppose a group member Alice needs to authenticate herself to another party Bob, who does not need to be a member of the group. The authentication protocol works as follows.

- Alice sends (E, t) to Bob.

- If the bit length of t is correct, Bob picks a random integer $r \in \{2, \ldots, n-1\}$, computes
$$E' = E^{tr} \pmod{n} \quad \text{and} \quad W = h(g^r),$$
where $h(x)$ is a one-way hash function, and sends E', W to Alice. Otherwise, Bob aborts the authentication.

- Alice computes $E'' = E'^s \pmod{n}$. If $h(E'') = W$, then Alice sends E'' to Bob; otherwise, she finds that Bob was cheating, and aborts the protocol.

- Bob checks if $E'' \equiv g^r \pmod{n}$, and accepts Alice as a valid group member if and only if this test succeeds.

The purpose of W is for Bob to prove that he knows the expected answer g^r, without giving away the answer. Since Bob knows g^r already, when Alice responds with g^r Bob learns no new information. If $h(x)$ is a true one-way hash function, then this prevents chosen message attacks.

4.3. Authenticated Key Exchange

The following procedure implements an authenticated version of Diffie-Hellman key exchange [9] based on our public key scheme, in which both Alice and Bob are group members.

- Alice and Bob exchange their public keys: (E_a, t_a), and (E_b, t_b), respectively.

- Alice and Bob pick random numbers r_a, r_b, calculate challenges $C_a = E_b^{t_b r_a}$, $C_b = E_a^{t_a r_b}$ and exchange challenges.

- Alice calculates $C_b^{s_a}$ which is g^{r_b} if all parties follow the protocols. Then Alice can get $g^{r_a r_b}$ which Bob can similarly obtain. Alice and Bob use this common value to derive a session key. Note that $g^{r_a r_b}$ is distributed over QR_n, which is only a subset of Z_n, so should be hashed to create a fixed-length symmetric key.

- Key confirmation step: Alice uses the session key to encrypt g^{r_b} using a symmetric encryption algorithm. Bob uses the session key to encrypt g^{r_a} using the same method. Alice and Bob exchange confirmation messages. If Alice/Bob find the decrypted value is equal to their own result, this finishes the key exchange protocol. Otherwise, the protocol is aborted.

In a real application, the key confirmation step can be integrated into the subsequent data transmission. Therefore, only two rounds of message exchange is needed for the authenticated key exchange protocol.

5. Security Properties of Proposed Scheme

We consider attacks when some group member keypairs may possibly be known, which supports the robustness condition of Definition 2.1. We abstract this as an attack model in which an attacker can collude with a set of legitimate parties, each with a legitimate keypair. A successful attack is one in which a new keypair is generated, with an identifier t that is valid and different from those of the colluding parties. The following theorem shows that, under the Strong RSA Assumption, it is intractable for an attacker to forge such a keypair.

Theorem 5.1 (Forgery-resistance) *If there exists a probabilistic polynomial time algorithm which takes a list of valid keypairs,* $(s_1, E_1, t_1), (s_2, E_2, t_2), \ldots,$ (s_k, E_k, t_k) *and with non-negligible probability produces a new keypair* (s, E, t) *such that* $E^{st} \equiv g \,(\mathrm{mod}\, n)$ *and* $t \neq t_i$ *for* $1 \leq i \leq k$*, then we can solve the flexible RSA problem with non-negligible probability.*

Proof: See the full paper [10]. ∎

Remarks. It is in fact possible to forge a group key with a duplicate t as long as the legitimate owner of the key with identifier t cooperates in the attack. However, such an attack is equivalent to key sharing, and if a group member is happy to let someone use its group member key, there is no way to prevent it. In real applications, if a group member key is found to be used by more than one party, we should assume it has been compromised, and revoke the identifier t, which would also revoke any forged keys with the same tag.

We next address the question of soundness of the authentication protocol. Loosely speaking, we show that, under the hidden exponent strong RSA assumption, only parties with a legitimate keypair (provided by the administrator) can succeed in the authentication protocol.

We first abstract an attacker of the authentication protocol as a pair of functions, which provide the two protocol messages the prover needs to provide. Given the public parameters g and n, function PublicKey(g, n) produces a purported public key (E, t) and (optionally) some private information p that it can use later (PublicKey can also use other publicly available information, such as the public keys of legitimate parties). Function Respond(g, n, E, t, p, c) takes the global public parameters $g, n,$ the information E, t, p produced by the PublicKey function, and a challenge c, and produces an answer a to the challenge. Note that an honest verifier creates a challenge as $c = E^{tr}$ for a random r, and accepts the prover's answer if and only if $a = g^r$ which is true if and only if (raising both sides to the $\log_g E^t$ power) $a^{\log_g E^t} = (g^r)^{\log_g E^t} = E^{tr} = c$.

Theorem 5.2 (Authentication Soundness) *If there exist probabilistic polynomial time algorithms PublicKey and Respond that succeed with non-negligible probability in fooling the verifier, then there exists a probabilistic polynomial time algorithm that solves the hidden exponent flexible RSA problem with non-negligible probability.*

Proof: Given an input (g, n, u) to the hidden exponent flexible RSA problem, we use the attacker's algorithms to create a solution as follows:

- Call PublicKey(g, n) and get (E, t, p).
- Call Respond(g, n, E, t, p, u) to get answer a.
- Set $v = a$ and $w = E^t$ and return (v, w) as an answer to the hidden exponent flexible RSA problem.

Note that if the attacker succeeds in providing a valid (E, t) and a, then as noted before the theorem we have $a^{\log_g E^t} = c$, or using the v and w notation we have $v^{\log_g w} = u$. Therefore, (v, w) is a valid solution to the hidden exponent flexible RSA problem if and only if the attacker succeeded in the authentication protocol. Since the latter occurs with non-negligible probability, by the condition of our theorem, the hidden exponent flexible RSA problem is solved with non-negligible probability. ∎

Note that while Theorem 5.2 guarantees that Alice can't cheat in the authentication, as explained in the previous section the use of the hash function guarantees that Bob can't cheat and extract additional information from Alice.

Finally, we note that our protocol exhibits a nice forward secrecy property. Specifically, if the administrator key is compromised, an attacker cannot use this knowledge to compute the private keys of the parties who have previously had keys created by the administrator. This is in contrast to recent work in Identity Based Encryption [6] where compromise of the administrator secrets results in compromise of all party's secret keys.

Theorem 5.3 (Forward Secrecy) *If the administrator secret (the factorization of* n*) is discovered at some point, and* σ *is large enough so that discrete logs are difficult to compute modulo a prime of* σ *bits, then all past and future key exchanges and authentications by previously authenticated parties are still secure.*

Proof: (Sketch) Recall that during uncompromised behavior, the administrator deletes any private keys that it had access to. Once the factorization of n is known, computing a secret key s from the public key (E, t) is basically a discrete logarithm problem (computing the $\log_{E^t} g$) in Z_p^* and Z_q^*. Since p and q are each at least σ bits long, and the condition in theorem requires that computing discrete logs modulo a prime of σ bits is intractable, then operations with previously authenticated keys remain secure even if the factorization of n is known. ∎

Note that since each of p and q need to be long enough to support a group with hard discrete log problems, if we want forward secrecy with a security level comparable to factoring a 1024-bit RSA value, then we need p and q to be 1024 bits, which makes the modulus 2048 bits.

6. Group Member Key Revocation

In case a group member leaves the group or is compromised by an attacker, the administrator should notify all the group members that a party holding a certain member key can not be regarded as a legitimate group member. In X.509 certificate based public key cryptosystems, the revocation is implemented by a "revocation list" which contains identifiers for all the revoked certificates. In our scheme, we revoke members by their tag, which is typically only 2 or 3 bytes, so revocation lists are quite compact. Therefore, the operations related to revocation list will use less system resources.

In dynamic settings, with many membership changes, revocation lists become large and cumbersome. To solve this problem, we propose an efficient method for re-keying all group members. In our re-keying technique, only group member public keys change, while the members retain their existing private keys without having to communicate them in any form whatsoever to the administrator. Thus, this operations is significantly less sensitive than the original key establishment step.

Our re-keying techniques works as follows: first, the administrator picks a random value $r \in \{2, \ldots, |QR_n|\}$ with GCD$(r, |QR_n|) = 1$, and then computes a new generator $h = g^r \pmod{n}$. Next, for each member that the administrator wants to keep in the group, the administrator computes a new public key E' from the existing public key E as

$$E' = E^r = (g^{s^{-1}t^{-1}})^r = (g^r)^{s^{-1}t^{-1}} = h^{s^{-1}t^{-1}} \pmod{n}.$$

These new public keys are either distributed to the group members or they can be published in a publicly accessible area.

This mechanism does not require any computation by the group member, and for a small group with at most a few thousand group members, it is a very simple task. Furthermore, since the key identifiers (the t values) remain unchanged, this technique can be combined with a revocation list for an efficient combined black-list/white-list group management system. The following theorem summarizes the security of our re-keying technique, and the proof is in the full version of this paper [10].

Theorem 6.1 *Under the Computational Diffie-Hellman Assumption over QR_n, no polynomial time probabilistic algorithm can compute an updated group member public key without knowledge of the administrator's private information (either the factorization of n or the secret value r).*

7. Conclusion

In this paper, we have presented an efficient public key scheme for authentication and authenticated key exchange for applications of embedded systems with a single trusted manager. We proved the security of the construction based on the Strong RSA and Computational Diffie-Hellman Assumptions. Finally, we introduced methods for key revocation and re-keying.

References

[1] S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In *Advances in Cryptology — Asiacrypt*, pages 452–473, 2003.

[2] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology*, pages 255–270, 2000.

[3] N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology — Eurocrypt*, pages 480–494, 1997.

[4] D. Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, pages 48–63, 1998.

[5] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $n^{0.292}$. *IEEE Transactions on Information Theory*, 46(4):1339–1349, 2000.

[6] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology — Crypto*, pages 213–229, 2001.

[7] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *Third Conference on Security in Communication Networks (SCN)*, pages 268–289, 2002.

[8] J. Camenisch and M. Michels. A group signature scheme based on an RSA-variants. Technical Report RS-98-27, BRICS, University of Aarhus, Nov. 1998.

[9] W. Diffie and M. Hellman. New direction in cryptography. *IEEE Transactions on Information Theory*, 11:644–654, Nov. 1976.

[10] H. Ge and S. R. Tate. Efficient authenticated key-exchange for devices with a trusted manager. Technical Report 2005-02, University of North Texas, Computer Privacy and Security (CoPS) Lab, 2005.

[11] M. Girault. Self-certified public keys. In *Advances in Cryptology — Eurocrypt*, pages 490–497, 1991.

[12] ITU-T. ITU-T recommendation X.509 – ISO/IEC 9594-8: Information technology – Open systems interconnection – The directory: Public-key and attribute certificate frameworks, 2001.

[13] H. Petersen and P. Horster. Self-certified keys — concepts and applications. In *Proc. Conf. on Communications and Multimedia Security*, pages 102–116, 1997.

[14] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology — Crypto*, pages 47–53, 1984.