Security Analysis and Vulnerabilities of TEGTSS-I Digital Signature Schemes

Project Work II. report

Beáta Éva Nagy

May 19, 2024

Supervisors:
Dr. János Tapolcai
Dr. Bence Ladóczki

1. Introduction

When constructing digital signature schemes, it is important to ensure the security of them. A possible attacker can be differentiated based on several criteria. Based on the attacker's knowledge, we can differentiate between *key-only attacks* (the attacker only has access to the public key of the signer), *known-message attacks* (the attacker has access to several valid message-signature pairs), *chosen-message attacks* (the attacker can choose messages and obtain their signatures), and the most advanced form, *adaptively chosen-message attacks* (the attacker can adaptively choose the messages to sign, based on the previously obtained signatures). Needless to say, if we ensure security against adaptively chosen-message attacks, it incorporates all the above as well, therefore, we will concentrate on such attacks [2].

The attacker's goal can vary as well: the strongest one is the *total break*, where the attacker obtains the secret key of the signer, allowing her to create valid signatures for any message. However, it is also an issue if the attacker "only" wants to forge the signature of the signer (without knowing the secret key itself), in the form of *universal, selective or existential forgery* (any message, specific messages, or at least one specific message can be successfully signed by the forger). To ensure the integrity of the signer, we need to be prepared for existential forgeries or any stronger attack. Therefore, we will examine the case of existential forgeries under adaptively chosen-message attacks [2].

The security of digital signature schemes is normally based on the computational hardness of given mathematical problems. We need to ensure that it is computationally challenging to perform existential forgery under adaptively chosen-message attacks. Many security proofs use the reduction method: they choose a computationally challenging problem, and show that forging a signature incorporates solving the reference problem, therefore, it is safe to assume that the probability of such event happening is negligible.

2. TEGTSS-I schemes

The so-called *El Gamal Type Signature Schemes* [1] are based on the algebraic properties of modular exponentiation and the discrete logarithm problem. Some well-known examples include the Schnorr signature scheme, the US-standard DSA, or the Korean-standard KCDSA, currently being in use for many applications.

As this group encompasses several signature schemes, and even variants of each scheme, the idea of trying to come up with a general security proof for this type of signature schemes arises. Firstly, building security proofs from scratch for every single variant, through utilizing their special characteristics is time-consuming and oftentimes complicated. Secondly, if a generalized proof exists for a certain type of signature schemes, then, if we construct a new scheme which falls into this category (satisfy all the required properties), we can accept its security without having to prove it individually.

One well-studied example of such generalization is the so-called *Trusted El Gamal Type Signature Scheme (TEGTSS)* [1]. The authors separate two types of TEGTSS schemes, based on the use of the utilized hash functions. They argue that practical digital signature schemes that fall into these categories are unforgeable relative to the discrete logarithm problem. Moreover, they utilize a *random oracle based model*, where given cryptographic hash functions are considered ideal random functions (random oracles), which means that the possible outputs are uniformly distributed on the

output set (deterministically), independent from the previous queries to the hash function. Also, in the model, a simulator is allowed to set the output of the random oracle to specific values, given that the input had not yet been defined. Other hash functions will require *collision-resistance*, which means that it is computationally impossible to find two distinct inputs that generate the same output [1] [3].

We will concentrate on the *TEGTSS-I* scheme in the following.

Our aim is to find a specific digital signature scheme, which satisfies (almost all) necessary properties of the TEGTSS-I scheme, but in reality, can easily be forged. We will construct an example of such scheme, then follow the unforgeability proof of the *TEGTSS-I* type schemes and prove that it applies to our scheme too. This contradiction leads to the idea that reducing to the discrete logarithm problem cannot prove unforgeability in every case, therefore, might be an incorrect assumption.

3. Modified Schnorr signatures

3.1. Original Schnorr signature scheme

As mentioned above, Schnorr's signature scheme [5] is a commonly used one, and is proven to be existentially unforgeable under chosen-message attacks in the random oracle model [4].

In general, the signature scheme utilizes two large primes p and q: $q|p-1$, $q \geq 2^{140}$, $p \geq 2^{512}$, and a *security number* t (in many cases, $t = 72$ or $t = 64$). Let g be a generator element of order q ($g^q = e$) in group $\mathbb{Z}_p^*$ (group of invertible integers mod p). Let x be the private key of the user, which is a random element in $\mathbb{Z}_q^*$, and its corresponding public version (public key) X, where $X = g^{-x} \mod p$ [5] .

To generate a signature, the user first chooses a random element r in $\mathbb{Z}_q^*$, which will be a secret, and computes its public version N (called the *nonce*), where $N = g^r \mod p$. Then, using a one-way hash function $H : \mathbb{Z}_q \times \mathbb{Z} \to \{0, \ldots, 2^t - 1\}$, the user calculates the hash of the message $h = H(N|m)$, and constructs the signature s of the message:

$$s = H(N|m) \cdot x + r \mod q = h \cdot x + r \mod q.$$

To verify the signature, one has to check whether

$$N == X^h \cdot g^s \mod p,$$

and check if

$$h == H(N|m).$$

3.2. Modified Schnorr signature scheme

Broadly speaking, the security of the original Schnorr signature scheme lies in the incorporation of the public nonce into the hash of the message, as the hash value depends on two parameters: the message itself, and a randomly generated value.

To prove this, we can construct a modified version of the Schnorr signature scheme, where the utilized hash function incorporates only the original message itself ($h = H(m)$). Formally, the signature will be:

$$s = H(m) \cdot x + r \mod q = h \cdot x + r \mod q.$$

One can easily prove that this slight modification results in decreased security, as the signature scheme becomes forgeable in the following way. The adversary can choose an arbitrary s' in $\mathbb{Z}_q^*$, then, as the hash function is publicly available, can calculate $h' = H(m')$, and from these, the corresponding nonce N' can be computed:

$$N' = X^{h'} \cdot g^{s'} \mod p.$$

Therefore, the adversary can successfully come up with a valid (s', N') pair, and thus a signature, for an arbitrary message m', without having to retrieve the secret key itself. Therefore, the modified Schnorr signature scheme is not secure.

4. Modified Schnorr signatures on TEGTSS-I

Next, we will compare the modified Schnorr signatures to the TEGTSS-I type signature schemes. According to the definition of TEGTSS signatures, two separate hash functions are used: H and G, where $q/2 < |H|, |G| < 2q$. H is the hash function of the message, and has to be collision-resistant, while G is the hash function of the nonce, which has to be an ideal random function (a random oracle). Therefore, instead of N, its hashed value, $R = G(N)$ will be publicly available.

Three functions are defined, F_1 represents the signature, while N is defined as $N = g^{F_2} \cdot X^{F_3}$. We will directly apply them to the modified Schnorr scheme:

- $F_1(r, x, h, R) = s \mod \text{q}$
- $F_2(s, h, R) = s \mod \text{q}$
- $F_3(s, h, R) = h \mod \text{q}$.

These three functions have to satisfy **three requirements** in order to be a TEGTSS-I signature scheme [1].

1. The requirement

   $$F_2(F_1(r, x, h, R), h, R) + x \cdot F_3(F_1(a, b, h, R), h, R) = r \mod q$$

   applies, because

   $$F_2(F_1(r, x, h, R), h, R) + x \cdot F_3(F_1(a, b, h, R), h, R) =$$

   $$= F_2(s, h, R) + x \cdot F_3(s, h, R) = s + x \cdot h = r \mod q$$

2. The second requirement: if $h_1 \neq h_2$ then $F_3(s_1, h_1, R_1) \neq F_3(s_2, h_2, R_2)$ is true by definition.

3. It is required that for a fixed verifying tuple (N, s_1, h_1, R_1) there is a one-to-one map between the values of R_2 and h_2 s.t. (N, s_2, h_2, R_2) verifies the TEGTSS equation and $F_3(s_1, h_1, R_1) = F_3(s_2, h_2, R_2)$.

   Since there is no connection between the values of R_2 and h_2 (R_2 solely depends on the random oracle, which is not predetermined, for example in the upcoming forking lemma, it

will be changed several times). The one-to-one map property only applies if the random oracle function is fixed, but in general, the property does not apply for the modified Schnorr scheme.

We can conclude that two of the three TEGTSS-I properties apply to the modified Schnorr scheme. We will examine where the third property is used in the security proof, and whether omiting this requirement changes the security results.

5. Security proof of TEGTSS-I

Next, we will investigate the proof of security of the TEGTSS-I scheme, applied to our modified Schnorr signature scheme.

The *forking lemma* is an important lemma used for the proof of security. Broadly speaking, it states that if an attacker uses an ideal hash function and can successfully construct a valid signature with non-negligible probability, then, the forking algorithm rewinds the attacker to the point before querying the random oracle (the ideal hash function). Then, it is possible to provide a different random oracle response, resulting in a second valid signature with non-negligible probability. From these two signatures, one can extract the secret key, meaning that this technique can be used to solve the underlying discrete logarithm problem [4].

The proof of TEGTSS-I is based on the improved version of the forking lemma, Lemma 5.1 [1].

Lemma 5.1 (The Improved Forking Lemma). *Let us consider a probabilistic polynomial time Turing machine $\mathcal{A}$, called the attacker, and a probabilistic polynomial time simulator $\mathcal{B}$. If $\mathcal{A}$ can find with probability $\varepsilon > 4/q$ a verifiable tuple (m, N, s, h, R) with less than Q queries to the hash function, for a new message m and for a R directly defined by G, then with a constant probability $1/96$, with $(1 + 24Q\ell\log(2\ell))/\varepsilon$ replays of $\mathcal{A}$ and $\mathcal{B}$ with different random oracles, $\mathcal{A}$ will output $\ell + 1$ verifiable tuples $(m_i, N_i, s_i, h_i, R_i)_{i=1,\dots,\ell+1}$ such that the R_i are pairwise distinct, and all the N_i equal for TEGTSS-I schemes.*

The main theorem, Theorem 5.2 argues about the security of TEGTSS-I type schemes [1].

Theorem 5.2. *Suppose that G is an ideal random function but H a collision-resistant hash function. Given an attacker $\mathcal{A}$ that can find with probability ε a verifiable tuple (m, N, s, h, R) for a new message m, with less than Q queries to the hash function G, then with constant probability $1/96$, with less than $25Q/\varepsilon$ replays of $\mathcal{A}$, with different random oracles, $\mathcal{A}$ extracts the secret key x.*

Extracting the secret key would imply solving the discrete logarithm problem, proving the impossibility of $\mathcal{A}$ finding such tuple with probability ε. In the following, we will prove that its proof applies to the modified Schnorr signature scheme too, implying that the security argument of Theorem 5.2 does not apply in every case.

In the proof of Theorem 5.2, when the attacker outputs a new verifiable tuple (m, N, s, h, R), there are two cases of extracting the secret key. In the first case, the simulator used by the attacker already produced a verifiable tuple (m', N, s', h', R') s.t. $m \neq m'$ (the simulator defined the value of $G(N)$, as the random oracle is programmable), therefore, due to collision-resistance, $h \neq h'$. Due to the second requirement of TEGTSS-I schemes, N is represented in two different ways (as F_3 and F_3' differ), from which, the extraction of the private key is possible, simply by subtracting the two signatures from each other:

$$s - s' = (h - h') \cdot x \mod q.$$

In the second case, the attacker directly queries G to get $G(N)$. The attacker uses the forking lemma on the verifiable tuple (m, N, s, h, R), and by successively querying G, after less than $(1 + 24Q)/\varepsilon$ replays of $\mathcal{A}$, gets a verifiable tuple (m', N, s', h', R') s.t. $R \neq R'$. It is important to emphasize that Lemma 5.1 only applies to TEGTSS-I type schemes. However, intuitively, it also applies to the modified Schnorr scheme, as the latter is a simpler signature scheme, as R is not utilized in the signature generation, only for hashing N. Therefore, the probability of finding a verifiable tuple (m', N, s', h', R') s.t. $R \neq R'$ cannot decrease , as N depends on one less variable.

To produce two distinct representations of N, and therefore extract the secret key, the original proof utilizes the one-to-one mapping between the values of R and h, arguing that due to the collision-resistant property of H, the probability that $F_3 = F_3'$ is vanishingly small. In our case, we have to investigate the probabilities of $F_3 = F_3'$ and $F_2 = F_2'$.

Similarly to the original proof, the probability that $F_3 = F_3'$ is true is vanishingly small, because this can only happen if $h = h'$, therefore $m = m'$. The inequality in the values of F_3 and F_3' implies the inequality in the values of F_2 and F_2' too, resulting in two different representations of R, and the extraction of the secret key.

6. Conclusion

During this project, we investigated the security of the TEGTSS-I type signatures which were proven to be unforgeable under an adaptively chosen-message attack in the random oracle model. We constructed a modified version of the Schnorr signature scheme, where the hash of the message only includes the message itself, making it vulnerable against forgeries. Then, we could see that this signature scheme almost falls into the category of TEGTSS-I type signatures, missing only a one-to-one map property. Even though this property is utlized in the security proofs, we can argue that since the modified Schnorr signature scheme is very simple, omiting this requirement, and utilizing the specific characteristics of the modified Schnorr scheme instead, the security results will not change.

This observation raises several questions. Firstly, whether we can actually reduce the problem of the security of TEGTSS signature schemes to the discrete logarithm problem, secondly, whether the use of the random oracle model alters the security results. Please note, that these results are prelimitary findings, and we are still working on making the arguments more precise, possibly by investigating the proof of Lemma 5.1 more thoroughly.

In the future, I would like to continue working on a more thorough security analysis, possibly by including other examples of practical or artificially constructed (vulnearable) signature schemes, and trying to apply the TEGTSS-I properties. Furthermore, I would like to extend my knowledge and investigate other security proofs that utilize the random oracle model, to investigate its role in the proofs.

References

[1] Ernest Brickell et al. “Design Validations for Discrete Logarithm Based Signature Schemes”. In: vol. 1751. Sept. 2001. ISBN: 978-3-540-66967-8. DOI: 10.1007/978-3-540-46588-1_19.

[2] Gianluca Dini. “Digital Signatures: Types of Attacks and Security Considerations”. In: *Network Security* (2023). Available at: http://docenti.ing.unipi.it/g.dini/Teaching/sanna/lecturenotes/applied-cryptography-digital-signature.pdf, pp. 13–17.

[3] Pascal Paillier and Damien Vergnaud. “Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log”. In: *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*. Vol. 3788. Lecture Notes in Computer Science. Springer, 2005, pp. 1–20. DOI: 10.1007/11593447_1. URL: https://iacr.org/archive/asiacrypt2005/001/001.pdf.

[4] David Pointcheval and Jacques Stern. “Security Arguments for Digital Signatures and Blind Signatures”. In: *Journal of Cryptology* 13 (Oct. 2001). DOI: 10.1007/s001450010003.

[5] Claus-Peter Schnorr. “Efficient Identification and Signatures for Smart Cards”. In: *Advances in Cryptology - CRYPTO ’89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Vol. 435. Lecture Notes in Computer Science. Springer, 1989, pp. 239–252. DOI: 10.1007/0-387-34805-0_22.