International Journal of Security and Its Applications
Vol.8, No.3 (2014), pp. 97-106
http://dx.doi.org/10.14257/ijsia.2014.8.3.11

Distributed Signing Protocol for Tamper-Resistant Module

Shinsaku Kiyomoto, Tsukasa Ishiguro and Yutaka Miyake
KDDI R & D Laboratories Inc.
kiyomoto@kddilabs.jp

Abstract

In this paper, we present a protocol for a ID-based signature scheme using a tamper-resistant module that holds a private key for signing. This is a distributed-signature scheme, and a computation using a private key is executed on the tamper-resistant module (TRM), with the remaining computations performed on a host PC. The scheme is secure against both passive and active adversaries, even those that can corrupt the host PC. The computational complexity on the TRM is reduced to one multiplication, one addition, and one symmetric key decryption, and the transaction data size is small.

1 Introduction

Leakage of secret information such as private keys from side-channel information has become a major security threat. Using such a technique, an adversary can obtain secret information from a client terminal that has been stolen or infected with malware by an adversary. A client terminal cannot be assumed to be a trusted platform, and we must consider a special security function so that secret information can be held securely. A trusted platform module (TPM) or a tamper-resistant module provides a secure computational environment, and can be used to improve system security. There are many such modules in commercial use, not only TPM but also SIM cards for mobile phones and other smart cards. However, some cryptographic primitives are difficult to execute on tamper-resistant modules, due to limited computational resources. Thus, we had to consider a way to reduce the computational complexity of the code to be executed on the tamper-resistant module. In this paper, we present a protocol for a ID-based signature scheme using a tamper-resistant module that holds a private key for signing. This is a distributed-signature scheme, and a computation using a private key is executed on the tamper-resistant module (TRM), with the remaining computations performed on a host PC. The scheme is secure against both passive and active adversaries, even those that can corrupt the host PC. The computational complexity on the TRM is reduced to one multiplication, one addition, and one symmetric key decryption, and the transaction data size is small.

The rest of this paper is organized as follows: Section 2 presents related work about protocols using tamper-resistant modules. A basic signature scheme proposed by Hess is discussed in Section 3 and our protocol that executes Hess's signature scheme is presented in Section 4. Section 5 analyzes the security of the proposed scheme and our concluding remarks are in Section 6.

ISSN: 1738-9976 IJSIA

2 Related Work

Digital signature schemes are a key function for system security and widely used for several applications [9, 11, 21, 23]. Tampering attacks are cryptographic attacks on the implementation of cryptographic algorithms [13], and successful tampering with a device will reveal secret information. There are many types of tampering attacks, which may use infrared radiation, incoming power, or clock timing signals as the delivery mechanism for hostile code[1, 5, 22]. Fault *et al.* proposed a countermeasure for one particular tampering attack [12]. Attacks on the physical memory can also obtain secret information. The goal of leakage-resilient cryptosystems is to remain secure even if arbitrary (but limited) secret information is leaked to an adversary. Katz and Vaikuntanathan proposed a signature scheme [19] that uses many secret keys to achieve leakage resilience. This existing approach sounds theoretically secure against certain tampering attacks, but appears to be impractical in the real-world environment. Using tamper-proof hardware is another approach to achieve security against side-channel analysis. The first use of tamper-proof hardware for cryptographic purposes was for theoretical consideration in software protection [14] and for E-cash [8, 10, 6]. Several security protocols [18, 7, 20, 15] for multi-party computation using tamper-proof hardware have been presented. The security of their protocols was discussed in the context of universal composability. Hazay and Lindell presented oblivious search protocols [16] using standard smart cards. In their setting, the smart card is assumed to be uncorrupted, and is modeled as tamper-proof hardware having on-board cryptographic operations and storing a secret key. Thus, their model is based on an actual smart card and makes realistic security assumptions for practical services.

In this paper, we consider an ID-based signature scheme using a tamper-resistant module in a practical setting and analyze its security.

3 ID-Based Signature Scheme

In this section, we introduce preliminary information and then present the ID-based signature scheme [17] that is used for our distributed signature scheme.

3.1 Bilinear Pairing

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let $g \in \mathbb{G}_0$ be the generator of $\mathbb{G}_0$ and e be a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. The bilinear map e has the following properties:

- Bilinearity: $\forall u, v \in \mathbb{G}_0, a, b \in \mathbb{Z}_p, e(u^a, v^b) = e(u, v)^{ab}$.
- Non degeneracy: $\forall u \in \mathbb{G}_0, e(u, u) \neq 1$

We make the following two computational assumptions as to [17].

Definition 1. (Computational Diffie-Hellman Problem) $\mathbb{G}$ denotes a group of a prime order p. Select $a, b \in \mathbb{Z}_p$ at random. For given g, g^a, g^b, the CDH is to compute g^{ab}.

Definition 2. (Bilinear Diffie-Hellman Problem) Given g, g^a, g^b, g^c for some $a, b, c \in \mathbb{Z}_p$, the BDH problem is to compute $e(g, g)^{abc}$.

The signature scheme is based on the above problems, which are computationally hard.

3.2 Signature Scheme

The ID-based signature scheme proposed by Hess [17] in 2003 consists of three steps as follows:

Setup: Let P be $P \in G_1$. The master secret is a random integer $s \in (\mathbb{Z}/q\mathbb{Z})^*$, and its public key is $P_{pub} = sP$. There exist two hash functions $H_1 : \{0,1\}^* \to G_1$ and $H : \{0,1\} \times G_2 \to (\mathbb{Z}/q\mathbb{Z})^*$.

Extract: For the public information (ID) $\mathrm{ID} \in \{0,1\}^*$, a public key is computed as $Q_{\mathrm{ID}} = H_1(\mathrm{ID}) \in G_1$, and its secret key is computed as $S_{\mathrm{ID}} = sQ_{\mathrm{ID}}$.

Sign : For computing the signature of a message $m \in \{0,1\}^*$, a signer uses his/her secret key S_{ID}, and an arbitrary $P_1 \in {G_1}^*$ and random integer $k \in (\mathbb{Z}/q\mathbb{Z})^*$ and executes the following steps:

1. $r := e(P_1, P)^k \in G_1$
2. $v := H(m, r) \in (\mathbb{Z}/q\mathbb{Z})^*$
3. $u := vS_{\mathrm{ID}} + kP_1 \in G_1$

The signature is the pair $(u, v) \in G_1 \times (\mathbb{Z}/q\mathbb{Z})^*$.

Verify : when the message m and the signature (u, v) are received, the verifier computes a signature Q_{ID} and then computes $r' = e(u, P)e(Q_{\mathrm{ID}}, -P_{pub})^v$. Finally, the verifier computes $v' = H(m, r')$ and compares it. If the pair (u, v) is correct, then $r' = e(u, P)e(Q_{\mathrm{ID}}, -P_{pub})^v = e(P_1, P)^k = r$. Thus, $v' = H(m, r) = v$.

4 Distributed Signing Scheme

In this section, we present a distributed signing scheme based on Hess's ID-based signature scheme. Step *Sign* is modified in our scheme, and the remaining three steps *Setup, Extract*, and *Verify*, are identical to Hess's scheme. The concept of a distributed signing scheme is that an important part of the signing computation using a private key is executed on a tamper-resistant module and the remaining parts are executed on a host PC. We assume that the user of the host PC (HPC) and the tamper-resistant module (TRM) share a password pw. The protocol is described as follows:

1. The HPC computes $r := e(P_1, P)^k \in G_1$, $v := H(m, r) \in (\mathbb{Z}/q\mathbb{Z})^*$, and kP_1. Then, the HPC requests that the user inputs a password pw, and the HPC computes an encrypted message $E = \varepsilon_{pw}(v||kP_1)$ using the symmetric encryption scheme ε. Note that the password pw is deleted from the internal state of the HPC immediately. The HPC holds v and m. The HPC sends E to the TRM.
2. The TRM decrypts E and obtains v and kP_1. Then, the TRM computes $u := vS_{\mathrm{ID}} + kP_1 \in G_1$. The TRM sends u to the HPC.

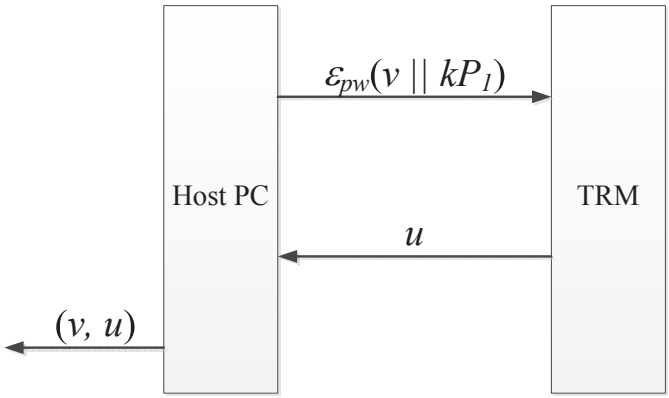

Figure 1. Signing Protocol

3. The HPC checks the validity of the signature data u. If u is not correct, the HPC terminates and outputs $\perp$; otherwise, the HPC outputs (v, u) as the signature of the message m.

The communication between the HPC and TRM is generally slow; thus, inputting the message m to the TRM via the communication channel is very time consuming. The above scheme only sends a hash value of the message. Thus, even though the message is long, the transaction time is short.

5 Security Analysis

In this section, we present a security analysis of the proposed scheme.

5.1 Tamper-Resistant Module

We assume that a tamper-resistant module (TRM) provides a high level of physical security, which is resistant to side-channel analysis as in [16], so it is assumed to be uncorrupted. Actually, it is impossible to ensure perfect security against any side-channel analysis in theory; but it is a reasonable assumption in practice if we use high-end smart cards satisfying FIPS 104-2, level 3 or 4 certification. The TRM is also a trusted entity. This means that the vendor of the TRM is trusted, and it is believed that the vendor did not incorrectly implement the TRM, intentionally or otherwise.

5.2 Adversary Model

The adversary model is defined using security models for cryptographic protocols [3, 4, 2]. The following queries are considered for the security analysis:

- **Execute**(H_i, T_j, m_l)**.** Let H_i and T_i be a HPC and a TRM respectively. This query is used to obtain all communication logs of a signature generation between H_i and T_i and the signature of message m_l.
- **Send**(X, d_i)**.** The **send** query is used to obtain the corresponding reply for transaction data d_i from an entity X.

- **Corrupt(H_i).** This query is used to obtain the internal state of the HPC H_i. Note that the query cannot be made to a TRM due to the nature of the tamper-resistance as in 5.1.

We assume two types of adversaries for the distributed signing scheme: a passive adversary and an active adversary. The adversaries can capture communications between the HPC and the TRM. The adversaries cannot intercept executions on the HPC but they may dump the contents of the physical memory of the HPC. The two adversary models can be defined to use the above queries as follows:

Passive Adversary. A passive adversary can select any message m and capture communications using an **Execute** query for the message m. We assume partial corruption of the HPC. The adversary uses **Corrupt** queries to obtain the internal state of the HPC during the distributed signing scheme. Note that the adversary cannot achieve any result by sending a **Corrupt** query to the TRM, and the user does not input to the HPC after the corruption.

Active Adversary. An active adversary can select a message m and can capture the communications and internal state of the HPC using an **Execute** query and a **Corrupt** query respectively. Furthermore, the adversary can send data to the TRM or the HPC using a **Send** query in order to obtain the corresponding reply from the TRM and the HPC.

We consider an active adversary who is stronger than the passive adversary in our security analysis.

5.3 Forgery of Signature

Forgery of signatures is one of the most significant attacks on signature schemes. Existential forgery is a specific attacker technique that is used to formally determine the security of digital signature schemes[1]. Given a verifying key, an existential forgery is achieved if the attacker finds a signature s for at least one new message m, such that the signature s is valid for m with respect to the victim's verifying key. In this analysis, we assume an active adversary who forges a valid pair comprising a message m and its signature $s = (v, u)$.

Theorem 1. An adversary $\mathcal{A}$ exists who makes at most q_e **Execute** queries, q_s **Send** queries, and q_c **Corrupt** queries. The success probability of existential forgery is $Pr[\mathcal{A}\ \ win] = Pr[\mathcal{A}_{SF}\ \ Succ] + Pr[\mathcal{A}_{\epsilon}\ \ Succ] + 2^{-|pw|} \cdot q_s$, where $Pr[\mathcal{A}_{SF}\ \ Succ]$, $Pr[\mathcal{A}_{\epsilon}\ \ Sccu]$, $|pw|$ are the success probability of signature forgery by an adversary $\mathcal{A}_{SF}$ in the original signature scheme, the success probability for recovering a plaintext by an adversary $\mathcal{A}_{\epsilon}$ from the ciphertext in the symmetric-key encryption scheme ϵ, and the length of the password pw.

Proof Sketch. In the simulation, the adversary can gather valid pairs comprising a message and its signature without the target message m and related information by using three queries. The point of the proof is whether an adversary can improve a success probability of an attack by using the above queries, in comparison with executing the original signature scheme.

The advantages of the three queries are discussed as follows:

[1]SpringerReference, `www.springerreference.com/`

ADVANTAGE BY USING **Execute** QUERIES. This query just simulates protocols between the HPC H_i and the TRM T_i; thus the adversary obtains a message m_i, its signature (v_i, u_l), and transaction data $\epsilon_{pw}(v||kP_1)$. The adversary achieves no gain from these data. There are two strategies for signature forgery: breaking the symmetric-key encryption scheme and password guessing. If the adversary can break the cipher or correctly guess the password pw, then the adversary obtains v and kP_1. Thus, the adversary can obtain a private key S_{ID} to compute $(u - kP_1)/v$, and generate a valid signature using the private key. We define the length of the password as $|pw|$ and assume the entropy of the password is high to avoid easy guessing of the password. Let the success probability of breaking ϵ (i.e. recovering plaintext) be $Pr[\mathcal{A}_\epsilon \ \ Succ]$. The total success probability of the attacks is $2^{-|pw|} + Pr[\mathcal{A}_\epsilon \ \ Succ]$. When we assume that the encryption algorithm ε is a random permutation and the password is difficult to guess within a practical time interval, the adversary cannot obtain any additional information from transaction data sent and received between the HPC and the TRM.

ADVANTAGE BY USING **Corrupt** QUERIES. Before step 1, an adversary obtains no information from using **Corrupt** queries, and only a valid signature (v, u) is obtained when the adversary uses a **Corrupt** query after step 2. Therefore, an adversary will try to use a **Corrupt** query between step 1 and step 2 in the distributed signing scheme. With a **Corrupt** query sent to the HPC, the adversary only obtains the message m and v as an internal state of the HPC. These are public values, so the adversary gains no advantage by using a **Corrupt** query.

ADVANTAGE BY USING **Send** QUERIES. Without revealing the password, the adversary has no meaningful probability of providing a correct input $\epsilon_{pw}(v||kP_1)$. The probability is $2^{-|\epsilon_{pw}(x)|}$ where $|\epsilon_{pw}(x)|$ is the length of the output of $\epsilon_{pw}(x)$. Even if the adversary can supply a correct input, the adversary only obtains its correct signature. However, the length of the input $|\epsilon_{pw}(x)|$ is generally much larger than the length of the password $|pw|$. Thus, queries that send guessed inputs do nothing to reduce the computational complexity of the attack. The **Send** query is also used for on-line guessing of the password pw. The adversary generates v and kP_1, then encrypts them using a guessed password pw and the symmetric-key encryption algorithm ϵ. The adversary obtains a signature (v, u) by sending a query to the TRM. If the signature is correct, the guessed password is correct. The probability of this attack is $2^{-|pw|} \cdot q_s$ where q_s is upper bound of the number of **Send** queries.

After correcting his information by using the above queries, the adversary generates a valid signature for the target message m without using the TRM (no query for the message is accepted). Let the success probability of the signature forgery in the original signature scheme be $Pr[\mathcal{A}_{SF} \ \ Succ]$. The success probability of the distributed signature scheme is:

$$\begin{aligned} Pr[\mathcal{A} \ \ win] &= Pr[\mathcal{A}_{SF} \ \ Succ] + 2^{-|pw|} + Pr[\mathcal{A}_\epsilon \ \ Succ] + 2^{-|pw|} \cdot q_s \\ &\approx Pr[\mathcal{A}_{SF} \ \ Succ] + Pr[\mathcal{A}_\epsilon \ \ Succ] + 2^{-|pw|} \cdot q_s \quad \square \end{aligned}$$

When the advantage $Pr[\mathcal{A}_\epsilon \ \ Succ] + 2^{-|pw|} \cdot q_s$ is negligible, then the security of the distributed signature scheme is identical to that of the original signature scheme. The security of the scheme mainly depends on the number of queries q_s; this requires an online

Table 1. Comparison with existing scheme

Scheme	Exec. on HPC	Exec. on TRM	Tran. Data	Succ. Prob.
Existing	-	2 **Mul.**, 1 **Add.**, 1 **Exp.**, 1 **Hsh.**, 1 **Ran.**	$\|m\| + \|v\| + \|u\|$	$Pr[\mathcal{A}_{SF}\ \ Succ]$
Proposed	1 **Mul.**, 1 **Exp.**, 1 **Hsh.**, 1 **Ran.**, 1 **Enc.**	1 **Mul.**, 1 **Add.**, 1 **Dec.**	$\|v\| + \|u\| + \|kP_1\|$	$Pr[\mathcal{A}_{SF}\ \ Succ]$ $+Pr[\mathcal{A}_{\epsilon}\ \ Succ]$ $+2^{-\|pw\|} \cdot q_s$

dictionary attack to supply passwords to the scheme. To slow down the attack, we consider adding a delay interval for each signing process. With the interval, we can reduce the total number of q_s possible in each time frame.

5.4 Other Attacks

In terms of practical use, we have to consider not only the forgery of a signature but also other attacks. We discuss the following two attacks:

- *DoS attack*: The adversary alters u or v during the protocol. This attack may still be possible, even though the ISO/IEC 7816 protocol provides message authentication between the host and the TRM. The host has to check the validity of u, when receiving it from the TRM.
- *Invalid use of the tamper-resistant module*: An adversary may try to use the TRM to obtain a correct signature for an arbitrary message. However, an adversary who does not know the password cannot use the TRM correctly. Thus, the complexity of this attack is reduced to the complexity of a password guessing attack. An offline guessing attack is impossible, because the input $E = \varepsilon_{pw}(v||kP_1)$ includes a random integer k and the adversary cannot obtain k. The adversary can only execute the on-line guessing attack. Thus, obtaining the correct password when the password space is large enough is inefficient.

5.5 Comparison with the Original Scheme

The comparison with the case that all computations are executed in a TRM is shown in Table 1. **Mul.**, **Add.**, **Exp.**, **Hsh.**, **Ran.**, **Enc.**, **Dec.** are denoted scalar multiplication, modular addition, modular exponentiation, hash computation, random number generation, symmetric key encryption, and decryption, respectively. Only three lightweight operations are required for the TRM in the proposed scheme. The data size m is generally much larger than the size of kP_1; thus, transaction data size between HPC and TRM is reduced. One drawback of the proposed scheme is that the security of the scheme depends of the strength of the password used for secure communication between the TRM and HPC.

6 Conclusion

In this paper, we present a distributed signature scheme. The scheme is secure against a signature forgery attack under the condition that the user's password is secure. The computational complexity on the TRM is reduced to one multiplication, one addition, and

one symmetric key decryption. The transaction data size is $|v|+|kP_1|$. The signature scheme is able to use several security applications such as user authentication, identification, and non-repudiation in a mobile environment. In future work, we will implement our scheme on actual devices and measure the transaction time of the scheme. We will also consider other protocols that use a TRM.

References

[1] Ross J. Anderson and Markus Kuhn. Tamper resistance: a cautionary note. In *Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, WOEC'96, pages 1–1. USENIX Association, 1996.

[2] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology ? EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer Berlin Heidelberg, 2000.

[3] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, CRYPTO '93, pages 232–249. Springer-Verlag New York, Inc., 1994.

[4] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: the three party case. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, STOC '95, pages 57–66. ACM, 1995.

[5] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology*, 14(2):101–119, 2001.

[6] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 302–318. Springer-Verlag, 1994.

[7] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for uc secure computation using tamper-proof hardware. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, pages 545–562. Springer-Verlag, 2008.

[8] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 89–105. Springer-Verlag, 1993.

[9] European Commission. *Digital Signatures: A Survey of Law and Practice in the European Union*. Woodhead Publishing, 1999.

[10] R.J.F. Cramer and T.P. Pedersen. Improved privacy in wallets with observers. In *Advances in Cryptology - EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 329–343, 1994.

[11] Ratna Dutta, Rana Barua, and Palash Sarkar. Pairing-based cryptographic protocols : A survey. In *IACR e-Print archive, 2004-12038*, 2004.

[12] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In *Automata, Languages and Programming*,

volume 6755 of *Lecture Notes in Computer Science*, pages 391–402. Springer Berlin Heidelberg, 2011.

[13] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In *Theory of Cryptography*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer Berlin Heidelberg, 2004.

[14] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[15] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *Proceedings of the 7th international conference on Theory of Cryptography*, TCC'10, pages 308–326. Springer-Verlag, 2010.

[16] Carmit Hazay and Yehuda Lindell. Constructions of truly practical secure protocols using standard smartcards. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 491–500. ACM, 2008.

[17] Florian Hess. Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer Berlin Heidelberg, 2003.

[18] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *Proceedings of the 26th annual international conference on Advances in Cryptology*, EUROCRYPT '07, pages 115–128. Springer-Verlag, 2007.

[19] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer Berlin Heidelberg, 2009.

[20] Tal Moran and Gil Segev. David and goliath commitments: Uc computation for asymmetric parties using tamper-proof hardware. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, pages 527–544. Springer-Verlag, 2008.

[21] Meziani Mohammed Sidi Mohamed El Yousfi Alaoui, Pierre-Louis Cayrel. Improved identity-based identification and signature schemes using quasi-dyadic goppa codes. In *IJAST*, volume 35, pages 83–92, 2011.

[22] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer Berlin Heidelberg, 2003.

[23] Swati Verma and Birendra Kumar Sharma. An efficient proxy signature scheme based on rsa cryptosystem. In *IJAST*, volume 51, pages 121–126, 2013.

