SCIS 2003 The 2003 Symposium on Cryptography and Information Security
Hamamatsu,Japan, Jan.26-29,2003
The Institute of Electronics,
Information and Communication Engineers

A Secure Testament Revealing Protocol

Kyusuk Han * Fangguo Zhang * Jongseong Kim * Kwangjo Kim *

Abstract— The testament is the message opened to the public after the message writer is gone. In this paper, after modeling the secure testament revealing protocol under three parties, the rich, the family and the lawyer, we propose a scheme for this protocol under the defined security requirements. Also we claim that this research will play one of typical practices in multiparty cryptographic protocols.

Keywords: Signature based on Poof of Knowledge, DLP, Security, Testament

1 Introduction

When a person passed away, he (or she) may leave the last message or the testament. In this testament, he writes about heirships and any important decision after he is gone. Many people prepare their testament before they are close to death, and keep it in secret not to be revealed before they die. However, some people can leave their testaments open to the public on handling their corpses, like eyes, heart, and so on. In general, many people want to keep the secret of their testament.

It will be a big problem when the rich passed away. If the rich has incredibly much money and many estates, there can be a struggle for distributing the estate by the family members and other relatives. We can easily imagine that heirs and heiress are not always in the family members. The rich can choose his friends as the heir or contribute his whole estates to any charity organization or the public society. In these cases, if the family members or any relatives know that the rich leaves the testament, then any trial of forgery of his testament, enforcement to be profitable to themselves will be possibly occurred. So, it is important not only to keep the message in secret, but also to keep even the fact that he wrote his testament in secret.

Suppose the rich has his lawyer. He may leave his testament to the lawyer, and the lawyer may keep it safely. But, because the lawyer has the rich's testament, the untrustworthy lawyer may reveal the testament at his own will. Since the rich doesn't want the case that his lawyer or friend opens his testament before he is gone, the rich wants to keep the message in secret even from his lawyer. There has to be a protection from the forgery or intentional revealing of the testament while the lawyer keeping it.

We can imagine the case that the lawyer or the family can't be satisfied with the content of the testament, when they reveals the testament relevantly. So it happens that they try to forge the testament.

In addition, we can also imagine the case that any adversary insists that he is an heir and wants to reveal the testament. So, identification is required. When the rich passed away, the lawyer declares that he (or she) is the escrow. Then, the family contacts the lawyer and the lawyer wants to identify them, since the lawyer does not know about the family (or heir), and can consider them as an adversary (who wants to see the testament or forge it.).

Our protocol is related to multiparty cryptographic protocol. There were many previous work about MCP, like secret sharing[3, 14], verifiable secret sharing[8] and MCP when most parties are honest[2]. Our protocol assumed that two parties are untrustful among three parties.

The rest of the paper is organized as follows: In Section 2, we show the generalized model of testament revealing. In Section 3, we describe definition and security requirement of this environment. In Section 4, we discuss preliminaries related to our protocol. In Section 5, we firstly propose a secure testament revealing protocol based on discrete logarithm problem[1]. Section 6 shows security analysis of the protocol. We make concluding remark and suggest further works in Section 7.

2 Model of Testament

There are three parties, the rich, the family and the lawyer. In our model, we assume that the rich does not trust both the family and the lawyer and that the rich is believed to be trusted. The rich will be gone someday worrying about the future after his death. So, he will prepare the testament. When the rich writes the testament, he leaves it to his lawyer (or escrow[1]) and his family (or heir). When the family and the lawyer receive the information of testament, the information are given to them as encrypted message. We assume that the family and the lawyer do not contact each.

After the rich passed away, the family (or heir) and the lawyer (or escrow) reveals the testament. At first, the lawyer declares that he is really the legal escrow.

* International Research center for Information Security (IRIS), Information and Communication University (ICU), 58-4 Hwaamdong Yuseong-gu, Daejeon, 305-732, Republic of Korea

[1] The definition of escrow is who keeps the information of the testament without knowing its content.

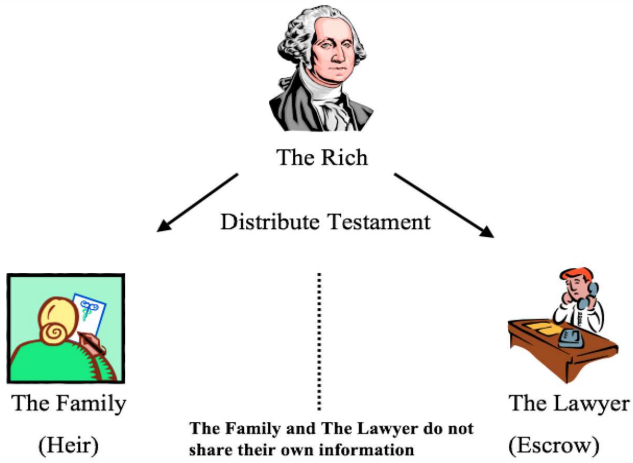

Figure 1: Model of testament - (The rich is alive.)

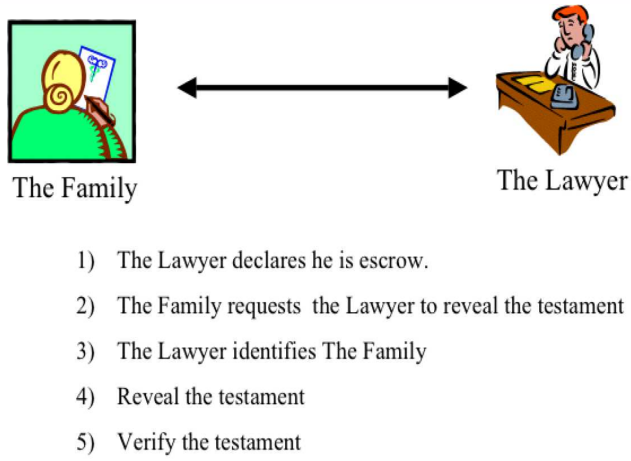

Figure 2: Model of testament - (The rich passed away.)

Then, the family requests the lawyer to reveal the testament. Then, the lawyer tries to check the identity of the family. If the lawyer checks the identity of the family, they reveal the testament. After that, they check the validity of the testament.

In the testament revealing, we can classify four steps, *generation*, *distribution*, *revealing* and *verification* as below.

- *Generation* - The rich generates his testament.
- *Distribution* - The rich sends his testament to the family and the lawyer.
- *Revealing* - The family and the lawyer reveal the testament together.
- *Verification* - The family and the lawyer check the testament's validity.

Since the fact that the testament was generated must be kept in secret, nobody should notice the distribution of the testament. The best way to keep it in secret that the rich sends the information to the family and the lawyer is to make the distribution looked as general sending an encrypted message. We can use well known cryptographic algorithms like RSA, ElGamal public key encryption in the distribution. Therefore we can omit *distribution* step. So, in our testament revealing protocol, we only deal with three steps: *generation*, *revealing* and *verification*.

3 Definition

We state the meanings of *testament*, *the rich*, *the family* and *the lawyer* in an intuitive way.

Testament is the last message of the rich. This original message is only known to the rich.

The Rich writes his testament, *i.e.*, generates the message and distributes his behaviors. Revealing is done without the rich. (the rich passed away.)

The Family can reveal the testament and request the lawyer to reveal the testament. They can check the validity of testament. In most cases the family are the heir.

The Lawyer can identify the family and check the validity of the testament. He keeps the testament without knowing about its content.

3.1 Risks in Testament Revealing

About the testament, there are several possible risks as follows.

(*Risk 1*) The testament should be kept in secret before it is properly revealed. When the family and the lawyer hold the encrypted testament, they should not decrypt the testament before the rich is gone. And, not even when the lawyer and the family keep it, it should be kept during the operation. Only when the operation is completed, the testament can be revealed.

(*Risk 2*) If the fact that the rich wrote the testament is known to the public, we can imagine any actions from adversaries like relatives who didn't receive the information of the testament. Sometimes, when the heirship of the rich is a big issue, the testament is written when the rich is still alive, then any action to the rich to know about the testament (e.g. to whom, how much, etc) can be possible, and the rich can be forced to modify the testament. Even the fact that the testament already exists should be in secret until the rich is gone.

(*Risk 3*) When the testament is revealed, the family knows about the testament. If the family doesn't satisfy about the testament, the family may try to forge it. Or, the untrustworthy lawyer could forge it while he was keeping it. So, conviction that the testament is not forged is required.

Additionally, we suggest a possible risk from the case that the lawyer and the family do not know each other when the rich is alive.

(*Risk 4*) When the lawyer declares that he is the escrow, there can exist any trial to pretend to be the family from adversary. So, the lawyer should be able to check the identity of the family.

3.2 Security Requirement

From the previous risks, we can consider the following requirements from the view point of security.

- *Confidentiality* - Testament should be kept in secret until the revealing is done.(Risk 1) And, even the fact that the testament exists should be kept in secret until the rich passes away. (Risk 2)
- *Unforgeability* - Testament should not be modified. (Risk 3)
- *Identification* - The lawyer should be possible to check the identity of the family. (Risk 4)
- *Message Authentication* - Both parties can check the validity of the testament.(Risk 2)

4 Preliminaries

We can design a secure testament revealing protocol is based on the interactability of the discrete logarithm problem.

4.1 Discrete Logarithm Problem

Let G be a finite cyclic group of order n. Let δ be a generator of G and let $\beta \in G$. *The discrete logarithm of β to the base δ*, denoted $\log_\delta \beta$, is the unique integer x, $0 \leq x \leq n-1$, such that $\beta = \delta^x$.

Given a prime p, a generator δ of Z_p^* and an element $\beta \in Z_p^*$, find the integer x, $0 \leq x \leq p-2$, such that $\delta^x \equiv \beta (mod\ p)$.

Discrete logarithm problem is known to be computational infeasible (or hard).

4.2 Hash function

To verify the integrity of message, hashing is generally used. Hash function has three potential properties for an unkeyed hash function h with input (x, x') and output (y, y'), those are *preimage resistance, 2nd-preimage resistance*, and *collision resistance*[9, 13].

- *preimage resistance* - for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, *i.e.*, to find any preimage x' such that $h(x') = y$ when given any y for which a corresponding input is not known.
- *2nd-preimage resistance* - it is computationally infeasible to find any second input which has the same output as any specified input, *i.e.*, given x, to find a 2nd-preimage $x' \neq x$ such that $h(x) = h(x')$.
- *collision resistance* - it is computationally infeasible to find any two distinct inputs x, x' which hash to the same output, *i.e.*, such that $h(x) = h(x')$.

But, in the testament, the size of message(*i.e.*, the testament) is really short. Assume the rich leaves the testament, denoted as m, and its hashed value, denoted as $h(m)$. In general, preimaging of h(m) is computationally infeasible(or hard). But in testament we can imagine it from guess attack. As example we can imagine that "I leave all my estate to my wife", or "my son", since the rich leave the testament with his related information. So, if we know how many family members he has, how much estate he has, then we can know preimage of $h(m)$. That is to say, though hash function has those potential properties, even with brute-force trial, preimaging of hashed testament is easier comparing with general hashing.

To solve this weakness we added a random number. For example, we assume a random number r, $1 \leq r \leq p$ where p is prime. Even the message m is short, knowing m from rm is difficult under modulo p. And, guess attack on $h(rm)$ is infeasible. With this we could solve this problem.

4.3 Signature based on a Proof of Knowledge

Camenisch and Michels showed the method proving the knowledge of the discrete logarithm, borrowing the notation from [4, 6], that is signature based on a proof of knowledge. They introduced a signature schemes derived from statistical (honest-verifier) zero-knowledge proofs of knowledge using the Fiat-Shamir heuristic[11, 12] and therefore called "signatures based on a proof of knowledge", SPK for short.

Following is the example of SPK: to prove the knowledge of the discrete logarithm of y to the base g and of a representation of z to the base g and h, and in addition, that the h-part of this representation equals the discrete logarithm of y to the base g. This is equivalent to the knowledge of a pair (δ, β) satisfying the righthand side of the following equation:

$$SPK\{(\delta, \beta) : y = g^\delta \wedge\ z = g^\beta h^\delta\}(m).$$

And, they also showed adoption of a protocol for showing the equality of two discrete logarithms given in [7] to the setting in which the group's order is unknown. We use this proving the equality of two discrete logarithm for verification.

(Definition 1) Let $e > 1$ be a security parameter. A pair $(c, s) \in \{0,1\}^k \times \{-2^{l_g+k}, ..., 2^{e(l_g+k)}\}$ Satisfying $c = H(g||h||y_1||y_2||y_1^c g^s||y_2^c h^s||m)$ is a signature of a message $m \in \{0,1\}^*$ with respect to y_1 and y_2 and is denoted as :

$$SPK\{(\delta) : y_1 = g^a \wedge y_2 = h^\delta\}(m).$$

5 Proposed Scheme

The rich generates his testament before he is gone. The distribution to *the family* and *the lawyer* was assumed to be executed safely.

After *the rich* passed away, *the family* will reveal the testament. *The family* requests *the lawyer* and *the lawyer* check if *the family* is valid. Then, *the lawyer* sends the encrypted testament to *the family*, and then *the family* reveals the testament. Finally, both parties check the validity of the testament, if it is valid, the testament has the legal document.

5.1 Notation

The following parameters are generated by *the rich*.
p : prime number. All computations are operated under modulo p throughout this paper.

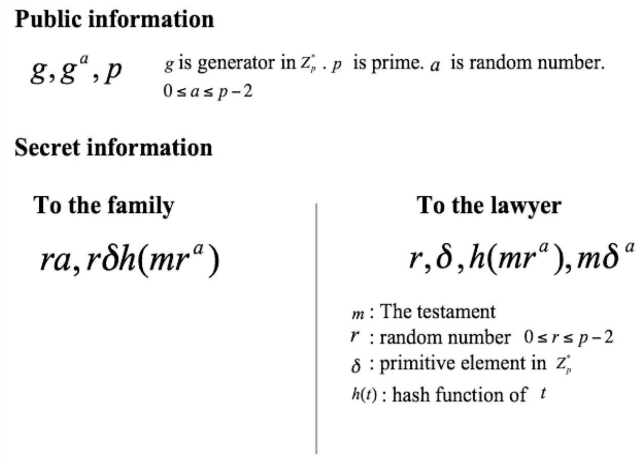

Figure 3: Testament generation

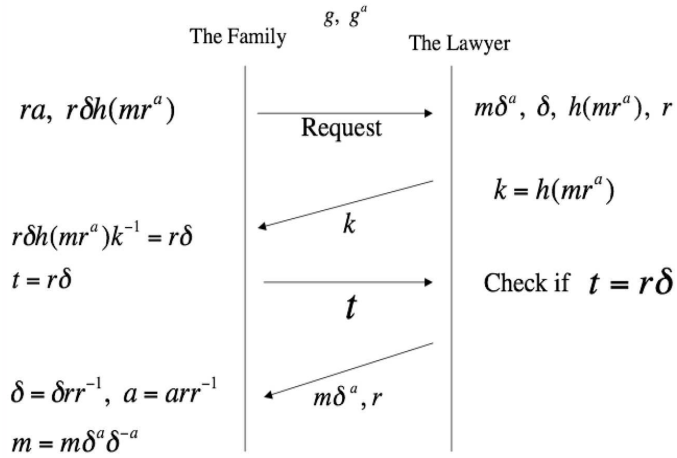

Figure 4: Testament revealing

δ : generator, $\delta \in Z_p^*$.
g : another generator in Z_p^*, which is a public information.
a : random number, $0 \le a \le p-2$.
r : another random number, $0 \le r \le p-2$.
m : message, which is testament.
F : information hold by *the family*
L : information hold by *the lawyer*

5.1.1 Generation

Generation follows the case as key generation of ElGamal public key encryption[10]. *The rich* selects two primitive elements g, δ in Z_p^*, and two random numbers, a and r, $0 \le a, r \le p-2$, p is prime. And he also selects a primitive element δ, $\delta \in Z_p^*$. *The rich* writes his testament, denoted as m.

After then, *the rich* generates as follows: ra, $r\delta h(m)$, $m\delta^a$, and $h(mr^a)$. And then *the rich* makes two group as follows:

$$F = \{ra, r\delta h(mr^a)\}, L = \{m\delta^a, \delta, r, h(mr^a)\}.$$

(g, g^a, p) will be opened to the public.

F and L can be considered as messages. So, we send them using conventional method. For example, we can use RSA public key encryption scheme, *the rich* encrypts F with *the family*'s public key and sends it to *the family*. Then *the family* can decrypts encrypted F and get the F. The channel between *the rich* and *the family* doesn't have to be secure or not. It's also the same with *the lawyer*. So, we omit the explanation of *the distribution*.

5.1.2 Revealing

When *the rich* is gone, *the family* and *the lawyer* can reveal the testament. After *the rich* passes away, *the lawyer* declares that he is the escrow of the testament to the public. *The family* insists that they are *the family*(or heir) of *the rich* and requests *the lawyer* to reveal the testament. To check the validity of *the family*, *the lawyer* sends k, which $k = h(mr^a)$ to *the family*. If *the family* is right, because only *the family* knows $r\delta h(mr^a)$, *the family* can reveal $r\delta$ from $r\delta h(mr^a)k^{-1}$.

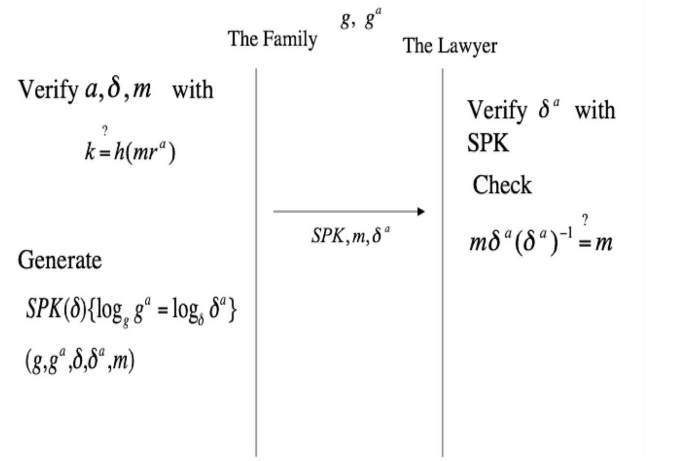

Figure 5: Testament verification

Then *the family* send $r\delta$ to *the lawyer*. Since *the lawyer* knows r and δ, *the lawyer* can compute $r\delta$ and compare with $r\delta$ from *the family*. With checking both $r\delta$, *the lawyer* can know the validity of *the family*. Then *the lawyer* send $m\delta^a$ with r to *the family*.

The family can reveal a and δ from ra and $r\delta$ with r. And, *the family* can reveal the message(*i.e.*, the testament) from $m\delta^a$ with a and δ.

5.1.3 Verification

The family check integrity of message m from $h(mr^a)$. If it is right, it's valid testament. Now, *the family* can believe the testament.

The lawyer also can check the validity. *The family* can generate δ^a because they know a and δ. They must show the equality of two discrete logarithm to *the lawyer*, with sending δ^a. Now, *the lawyer* knows g, g^a, δ, δ^a, and $m\delta^a$. So, with the property $\log_\delta \delta^a = \log_g g^a$, they can verify validity of δ^a, and then they can reveal the message m from $m\delta^a$. And they can know the validity of the message.

6 Security Analysis

6.1 Confidentiality

The family and *the lawyer* cannot reveal the testament by themselves, since they do not know about a. Though *the family* has ra, knowing r and a from ra is hard under modulo p, when p is prime.

Based on DLP, even δ is known, finding a from δ^a and δ is difficult[1]. So, *the lawyer* cannot reveal the message alone, though he knows $m\alpha^a$ and α.

The lawyer has $h(mr^a)$. Though *the lawyer* knows r, guessing preimage of mr^a is computationally infeasible, due to DLP and the property of hash function.

During revealing, though $h(mr^a)$, $r\delta$, $m\delta^a$ and r are known to any malicious adversary, it is impossible to know m from those four, since ra is never sent, and it is secure based on discrete logarithm.

And, when *the rich* generates the original testament, he divided all parameters in two, and only leaves g, g^a, and p opened to the public. But, it is the same as any public key encryption algorithm based on DLP. (Especially, it can be considered as public key of ElGamal public key encryption[10].) So, two group F and L can be sent as a 'message', and they are not recognized as testament.

So, the confidentiality is satisfied.

6.2 Unforgeability

If *the family* is untrusted, since *the family* can finally reveal the testament, and when the content of the testament is not profitable to *the family*, *the family* can try to forge the testament.

But, because *the lawyer* can check the equality of two discrete logarithm problem with the signature based on SPK, it is impossible to forge the original testament. *The family* must show the equality of two discrete logarithm problem, (g, g^a) and (δ, δ^a). So, *the family* must send $SPK\{(\delta) : log_g g^a = log_\delta \delta^a\}$
$(m, g, g^a, \delta, \delta^a)$, m and δ^a to *the lawyer*. Since *the lawyer* already knows g, g^a and δ, the lawyer can verify the validity of the testament with SPK.

On the contrary, we can assume that *the lawyer* is untrustful. Though *the lawyer* forge the $m\alpha^a$ or r, *the family* can know it, since *the family* can know $h(mr^a)$. It is based on the property of hash function[6]. Knowing r^a is really hard based on discrete logarithm problem. And preimage of $h(mr^a)$ is also impossible[6].

So, the unforgeability is satisfied.

6.3 Identification

When *the lawyer* declares that he is the escrow of the testament, any party can insist that they are *the family* and reques *the lawyer* to reveal the testament.

So, *the lawyer* can check from received parameter $r\delta$. *The lawyer* can check $r\delta$ with known r and δ. With this, *the lawyer* can convince the validity of *the family* from $r\delta$.

But, even the malicious adversary succeeds to get δ by chance, it needs the same computation time to get the validity with any other identification scheme, that the trials are p times(All computations are done under modulo p). If p is really big prime integer, the provability of success of the adversary gets really low.) And though he gets the validity from *the lawyer*, and get $m\delta^a$ and r, this case is not significant. The probability is just $\frac{1}{p}$. When p is big, it is negligible.

So, the identification is satisfied.

6.4 Message Authentication

Because only *the rich* knows the original testament, *the family* and *the lawyer* can only believe that the revealed testament is the same as the original.

So, message authentication should be done by both parties. *The family* should be possible to verify the testament from $h(mr^a)$, since *the family* finally knows a, r, and m.

For the verification of testament by *the lawyer*, we brought proving with showing the equality of two discrete logarithm from [5]. *The lawyer* verifies from the signature based on SPK. Because *the lawyer* already knows g and g^a, and he can get δ from revealing protocol. *The family* sends *the lawyer* $SPK(m)\{log_g g^a = log_\delta \delta^a\}$ that is proof that the family has valid a and sends valid δ^a. Because *the lawyer* has $m\delta^a$ with SPK, if he gets valid δ^a, then he can reveal m from $m\delta^a\delta^{-a}$. So, *the lawyer* can verify the testament.

Both two parties can check the revealed testament.

All security requirements are satisfied in our proposed testament revealing protocol.

7 Concluding Remark

We initiated to construct a secure testament revealing protocol as a one of typical practices of multiparty cryptographic protocols.

The model of testament consists of three parties, *the rich* who generates *the testament*, *the family* and *the lawyer*, who get the testament and reveal it.

Our model consists of three steps, *generation*, *revealing* and *verification*. *Distribution* of *the testament* is assumed to be executed easily under conventional method and omitted in our model. In the testament, guarantee of *confidentiality*, *unforgeability* and *message authentication* are most important. We could guarantee *confidentiality* based on discrete logarithm problem, DLP, *unforgeability* with DLP and potential properties of hash function, *message authentication* with properties of hash function and signature based on SPK. So, we made our protocol satisfying these requirements, and we tried to satisfy another security requirement, that is *identification*. This protocol is provably secure based on DLP.

We didn't assume the case of the collusion between the family and the lawyer in this paper. In practice, it is possible that they collude for the testament. As the future work, we need to improve this testament revealing protocol to prevent this assumption efficiently.

The research on this revealing protocol is just initiated. More clear formalization is required. Furthermore, extending to the random oracle model or implementation based on elliptic curve cryptography will be

possible. Also, we think that applying more improved idea like coin flipping for the identification.

We introduced a new model of cryptographic protocol on testament revealing. We don't think this is the optimal approach and expect many various approach on the secure testament revealing protocol.

References

[1] 'The discrete logarithm problem", C. Pomerance, editor, Cryptology and Computational Number Theory, volume 42 of Proceedings of Symposia in Applied Mathematics, 49-74, American Mathematical Society, 1990.

[2] M.Ben-Or, S.Goldwasser, and A.Wigderson, "Completeness theorems for fault-tolerant distributed computing.", In Proc. 20th ACM Symp. on Theory of Computing, pages 1-10, Chicago, 1988, ACM.

[3] G.Blakley, "Safeguarding cryptographic keys", In Proc. AFIPS 1979 National Computer Conference, pages 313-317, AFIPS, 1979.

[4] J.Camenisch, "Group signature schemes and payment systems based on the discrete logarithm problem", PhD Thesis, ETH Zurich, 1998, Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz.

[5] J. Camenisch and M. Michels, "A group signature scheme based on an RSA-variant", BRICS RS-98-27, ISSN 0909-0878, November 1998, *BRICS, Department of Computer Science University of Aarhus, All rights reserved.*

[6] J.Camenisch and M. Stadler, "Efficient group signature schemes for large groups.", In B. Kaliski, editor, Advances in Cryptology - CRYPTO'97, volume 1296 of Lecture Notes in Computer Science, pages 410-424, Springer Verlag, 1997.

[7] D. Chaum and T.Pedersen, "Transferred cash grows in size.", In R.A.Rueppel, editor, Advances in Cryptology-EUROCRYPT'92, volume 658 of Lecture Notes in Computer Science, pages 390-407, Springer-Verlag, 1993.

[8] B.Chor, S.Goldwasser, S.Micali, and B.Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults." In Proc. 26th IEEE Symp. on Foundations of Comp. Science, pages 383-395, Portland, 1985, IEEE.

[9] I. Damgard, "Collision free hash functions and public key signature schemes", Advances in Cryptology-EUROCRYPT '87 (LNCS 304), 203-216, 1988.

[10] T.ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory, 31 (1985), 469-472 .

[11] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity", Journal of Cryptology, 1:77-94, 1988.

[12] A. Fiat and A. Shamir. "How to prove yourself: Practical solution to identification and signature problems.", In A.M. Odlyzko, editor, Advances in Cryptology-CRYPTO'86, volume 263 of Lecture Notes in Computer Science, pages 186-194, Springer Verlag, 1987.

[13] R.C. Merkle, "Secrecy, authentication, and public key systems", UMI Research Press, Ahn Arbor, Michigan, 1979.

[14] A. Shamir, "How to share a secret", Communications of the ACM, 22:612-613, November 1979.