

WANTED: A Theft-Deterrent Solution for the Pervasive Computing World

Patrick Droz, Ceki Gülcü and Robert Haas

IBM Research, Zurich Research Laboratory

Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland

E-mail: {dro, cgu, rha}@zurich.ibm.com

Abstract— This paper presents a new theft-deterrent system called WANTED¹. The system relies on credits and blacklists, with each protected device periodically requesting new credit in order to continue operating. The main features of WANTED are resistance to attacks, guaranteed privacy, and scalability. The client part is a hybrid system consisting of an untrusted software component and a trusted hardware component. This offers maximum flexibility, while keeping the hardware requirements low.

I. INTRODUCTION

Portable high-tech devices tend to be valuable, therefore attractive candidates for theft. A variety of theft-protection methods already exist, see Fig. 1. They can be combined to increase the level of protection. Wireless signals can be used to prevent devices equipped with such transmitters from being removed from protected premises [1]. In contrast to physical locks, tracking is transparent to the user. Passive tracking systems, such as serial number registries, are easy to install, at least on a small scale. They act as deterrent against placing stolen devices on the open market. Active tracking offers a more aggressive stance by continuously monitoring the status of devices and potentially activating recovery procedures.

The need for effective theft-protection mechanisms will undoubtedly increase with the advent of pervasive computing.

In this paper, we present a novel theft-deterrent mechanism called WANTED, which belongs to the active-tracking category. WANTED assumes that theft-protected devices are at least occasionally connected to a network.

A theft-protection system, if it is to serve as a deterrent and be widely deployed, cannot be kept secret. Several software-only solutions exist today [2-4], but these can be removed or disabled by an astute adversary. Once a patch bypassing the software-only protection is available, it will be widely disseminated in a short period of time. On the other hand, hardware-only protection systems suffer from high complexity and price. For these reasons, we propose to use tamperproof cryptography hardware, albeit with minor enhancements, together with control software on the theft-protected device. Active tracking systems based on challenge-response authentication and secret-key cryptography are proposed in [5] and [6]. Being secret-key-based, these systems probably are not suitable for large-scale deployment.

The paper is structured as follows. Section II presents an overview of the protocol, the hardware requirements, and a

¹WANTED stands for Web-attached Network-device Theft Explorer and Deterrent.

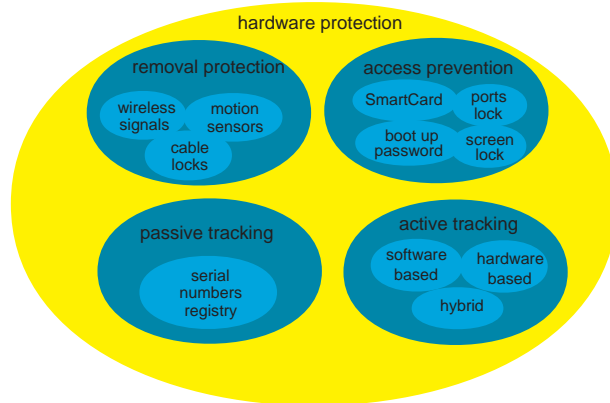


Fig. 1. Overview of theft-deterrent methods.

sample implementation. Section III addresses various attacks as well as privacy and scalability issues.

II. SYSTEM DESCRIPTION

The WANTED active-tracking theft-deterrent relies on the following assumption: neither the network nor the operating system of the client are trusted. In fact, the only trusted parties are the cryptography hardware integrated into the theft-protected device and, of course, the server involved in the theft-deterrent communication protocol, as shown in Fig. 2.

Communication between these two trusted parties is enabled by software running on the theft-protected device, and serves several purposes: one is to allow activation or deactivation of the theft-deterrent protection, the second is to run a credit-refresh protocol. Optionally, additional information can be exchanged once recovery procedures have been started.

Proof of property is required when a device is declared stolen. This is necessary to avoid denial-of-service attacks against legitimate users. For that purpose, trusted registries or signed proofs of property can be used. Also, when activation or deactivation of the theft-deterrent protection is requested by a user, proof of property should be presented. Management of trust, users, and theft-protected devices is an important issue [7, 8], and subject to future work. This paper focuses on the credit-refresh protocol presented below.

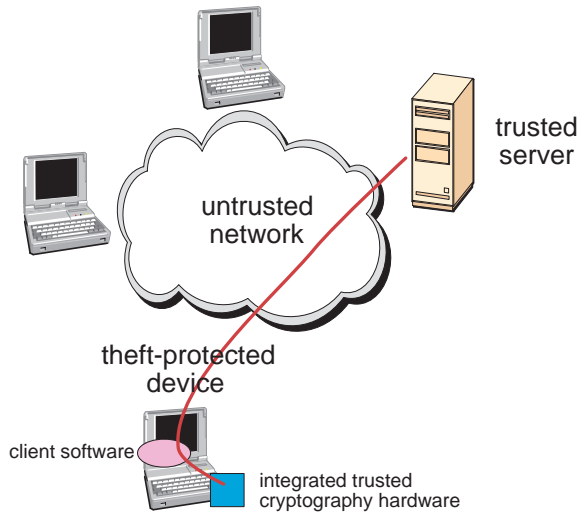


Fig. 2. System components.

A. Credit-Refresh Protocol Description

The WANTED credit-refresh protocol is the central piece of the theft-deterrent mechanism. The theft-protected device obtains signed tokens from a trusted server. These tokens contain “credit,” which allows the device to operate normally for a certain amount of time. The theft-protected device can ask for fresh tokens before the current credit expires. If the credit is exhausted the machine is no longer allowed to operate, except to allow the retrieval of new credit.

On the theft-protected device, the WANTED system consists of two components: a *trusted* hardware component and an *untrusted* software component. The software component is responsible for contacting a trusted server to obtain fresh credit and feed the results into the hardware component. The hardware component allows normal operation of the device if and only if the credit obtained from previous tokens has not yet expired.

The trusted server can be managed by any appropriate entity, which need not necessarily be the current user of the theft-protected device.

The trusted hardware component needs to possess the following set of capabilities:

1. to securely store the public key of the trusted server,
2. to verify signatures made by the trusted server,
3. to compute secure hashes,
4. to encrypt with the public key of the trusted server,
5. to securely store and run a small program and associated variables,
6. to securely store a unique ID for the protected device, called the Wanted-ID, denoted \mathcal{W} ,
7. to generate unpredictable nonces,
8. to shutdown the machine or perform other annoying operations², and

²The type of available operations depends on how the cryptography hard-

9. to generate time-based events³ using an independent oscillator.

Here, secure storage means that the expense of modifying securely stored data is higher than the benefits gained from a successful modification [9, 10].

The credit counter is decremented by one unit at each clock tick of the internal oscillator or each time a specific event occurs. As mentioned above, the hardware component will shut the machine down if the credit counter reaches zero or becomes negative.

Even if the credit count is zero or negative, the hardware component will let the device restart and run for a limited amount of time, called the *grace period*, that is long enough to allow the software component to obtain a fresh token.

The exact value of the grace period is device-dependent. For general-purpose computing devices, this should be long enough for the operating system to boot and fetch credit from the server. For most devices and operating systems, ten minutes should be ample but too short to really benefit from the device.

To locate and hopefully recover a stolen device, additional data could be transmitted to the server before the device shuts down. These data could include registry information, routing topology, and GPS data among others. Note that the attacker can remove the software component to prevent tracing.

A valid token is fed to the hardware component by the software component. At a safe margin prior to the expiry of credit, the software component invokes the `newQuery` method of the hardware component. This method returns a string, $E(\mathcal{W}||n)$, where n is a nonce generated by the hardware component, \mathcal{W} is the Wanted-ID and E denotes public-key encryption [11] with the server’s key. See Table I for the notational conventions used throughout this paper.

TABLE I
NOTATION SUMMARY

Notation	Explanation
$m_1 m_2$	concatenation of m_1 and m_2
$m_1 \oplus m_2$	binary XOR of m_1 and m_2
$E(m)$	encryption of m with the trusted server’s public key
$\text{sign}(m)$	Trusted server’s signature on m
$H(n)$	Secure hash of n

The software forwards this string to the trusted server. Whenever the server replies, the software component forwards the reply to the hardware component by invoking the `newToken` method.

The server’s reply consists of a signature⁴ on $U = \mathcal{W} \oplus H(n)$ concatenated with new credit, denoted by c . A secure hash is applied by the server on n [12]. If the Wanted-ID is blacklisted,

ware is integrated into the theft-protected device.

³Other types of events could also be considered.

⁴By convention signatures also include the signed message in clear.

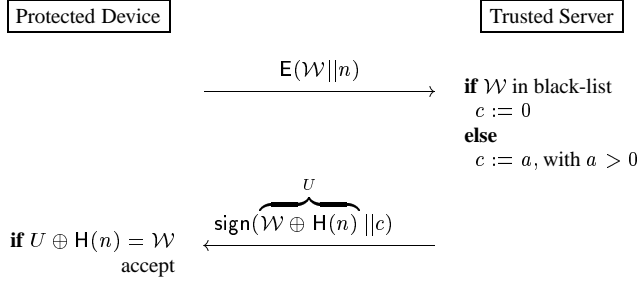


Fig. 3. Credit-refresh protocol.

i.e., the device was reported as stolen, then the value of c is zero. Otherwise, it is a positive integer a , whose exact value is policy-dependent. The exchange between the theft-protected device and the trusted server is summarized in Fig. 3.

The `newToken` method will accept a new token if it bears a valid signature, and checks whether $U \oplus H(n) = \mathcal{W}$, where n is the nonce generated by the previous call to `newQuery`. This check ensures that an adversary cannot replay previously sent tokens. The credit counter is set to c .

The Wanted-ID, \mathcal{W} , is revealed only to the trusted server. Assuming n is of the same bit length as \mathcal{W} , XOR-ing is secure against any adversary even with infinite computing power. Neither a computationally-bounded adversary nor the software component can retrieve \mathcal{W} from $E(\mathcal{W}||n)$ without knowing the trusted server's private key.

New credit c is an integer that could be interpreted as expressing absolute time or relative time. The absolute-time interpretation provides for a concise syntax to express policies, such as “work until October 1st, 2000.” However, the hardware component would require date-keeping capability, which is a rather complicated and expensive operation.⁵

Instead, we prefer the relative-time interpretation, in which the hardware component need not have a notion of date but simply decrements the credit counter at each clock tick. In the remainder of this document, the relative-time interpretation is assumed.

B. Credit-Refresh Protocol Implementation

The pseudo-code below presents a sample implementation of the code that runs on the hardware component. The number of instructions is very small, allowing most cryptography hardware to support such a protocol.

In this example, we assume that an oscillator is available. The code could be adapted to accommodate other types of events. The code is written in pseudo-code akin to the C language, and declares a few variables and three functions, namely `clockTick`, `newQuery` and `newToken`.

The `clockTick` function is called at each clock tick, assumed

⁵We cannot rely on the system clock, as it can easily be manipulated by the adversary.

```

1  uint128 wantedID;
2  int32 credit;
3  uint128 pendingNonce;
4  int32 gracePeriodCounter = 0;
5  //GRACE_PERIOD is set to 10 min
6  int32 const GRACE_PERIOD = 600;
7
8  // Called at each clock tick (i.e. each second).
9  void clockTick() {
10     credit--;
11     // Check for credit only after the grace period has elapsed.
12     if (gracePeriodCounter < GRACE_PERIOD)
13         gracePeriodCounter++;
14     else if (credit <= 0)
15         blockMachine();
16 }
17
18 // Called by the software component.
19 char* newQuery() {
20     if (pendingNonce == NULL)
21         pendingNonce = generateNonce();
22     return E(wantedID, pendingNonce);
23 }
24
25 // Called by the software component.
26 void newToken(char* m, BigInteger signature) {
27     uint128 U = *((uint128*) m);
28     if ((U ^ secHash(pendingNonce)) == wantedID) {
29         if (isValidServerSignature(m, signature)) {
30             pendingNonce = NULL;
31             int c = *((int32*) ((uint128*) m+1));
32             credit = c;
33             // Check whether device is stolen
34             if (c <= 0)
35                 blockMachine();
36         }
37     }
38 }

```

to be at intervals of one second each. This function first decrements the amount of available credit. If the grace period has not yet elapsed, it increments the grace period counter. Otherwise, it proceeds to check whether the amount of available credit is less than or equal to zero. If no credit is left, the machine is shut down.

The `newQuery` function returns the result of the encryption of the current nonce and the Wanted-ID. If the software calls `newQuery` before feeding the server's response, we reuse the previous nonce. This is *required* to prevent the cumulative token attack described in Subsection III.A.2. Incidentally, this also keeps the memory footprint small because never more than one nonce has to be remembered.

The `newToken` is called by the software to feed the answer of the server to the hardware component. This function takes a message m and a signature on the message. The code assumes that m is composed of a 128-bit unsigned integer U , followed by a 32-bit integer c . We use C-style pointer arithmetic to retrieve these values.

On line 28, we check whether the server's purported answer XOR-ed with the secure hash of the nonce matches our Wanted-ID. This check prevents replay attacks. Only if this inexpensive check succeeds do we proceed to verify the signature, a relatively expensive operation.

On line 30, when it is clear that the token is signed and fresh, the `pendingNonce` is reset. On the next two lines, we retrieve the value of c from the message and set the credit counter. A zero value of c indicates that the device is black-listed and needs to be shut down.

III. ANALYSIS

A prototype implementation of the WANTED system is under development. This will enable us to refine certain aspects further, and address problems omitted in this paper. Here, we discuss four different types of attacks [13], and how they can be countered. We elaborate also on the privacy issue, and explain why it is preserved. Finally, as for any potentially widely-deployed system, we need to address the scalability issue. We show that WANTED allows a single server to handle a very large number of clients.

A. Security

WANTED is based on two trusted parties: an independent server and the hardware component running on the theft-protected device. All other parties are untrusted. It is also assumed that the two trusted parties can readily communicate. While formal verification of cryptographic protocols [14, 15] can identify unexpected weaknesses in complex protocols, we consider that in the present case an informal verification is suitable [16].

A.1 Denial of Service

An adversary connected to the same network can choose to disrupt all communications between the device and the server or only the WANTED-related flows. If the disruption is sufficiently long, the device will eventually run out of credit and stop working.

Needless to say that this attack will not go unnoticed and that countermeasures can be taken to track down the disrupter.

In a highly centralized architecture where a few servers serve millions of clients, a denial-of-service attack can create havoc.

Another critical system, the Domain Name System (DNS), relies on 13 root servers, called A through M. If all 13 root name servers go down or become disconnected for an extended period of time, all name resolutions on the Internet would fail. The root servers happen to span multiple continents such that simultaneously disconnecting all 13 is highly unlikely.

As a straightforward extension, one could use multiple trusted servers per theft-protected device or use anycast addresses [17] rather than a single server. This would reduce the possibility of failure due to a broken server.

Alternatively, the trusted servers need not manage a huge number of devices but only devices within a well-defined administrative domain. A failing server would then only affect the devices within its domain.

WANTED is compatible with both architectures.

More sophisticated solutions using tamper-proof software [18], together with authentication of results from the hardware component, would prevent denial-of-service attacks originating from the protected device itself. This can for instance be the case when a virus prevents credits from being renewed.

A.2 Cumulative Token Attack

After a theft has occurred, there will be a certain delay before the legitimate owner can notify the trusted server of the theft. During this time the adversary may attempt to accumulate credits before the device appears on the blacklist, and then disconnect the device or simply remove the WANTED software component. The implementation presented in this paper thwarts this possibility, because

1. the nonce n is unpredictable,
2. there is only a single nonce n pending at any time. Until a valid reply is fed into it, the WANTED hardware component uses the same nonce n , and therefore returns the same $E(\mathcal{W}||n)$ whenever asked by the software component, and
3. credits are not cumulative. Whenever a new credit has been received and successfully verified, the current value of the credit counter is overwritten by the value of the newly received credit.

A.3 Token Deadlock

WANTED assumes that the theft-protected device and the server can communicate, i.e., that the operating system, the TCP/IP stack, and the software component are properly configured and functional on the theft-protected device.

Imagine that repairing a failure on the protected device lasts longer than the grace period of 10 min and there is no available credit. Then, the hardware component will shut the machine down before the device is restored to working order. To obtain a fresh token requires a running device, but repairing the device requires a token. This is a deadlock situation.

As such a deadlock situation will render a machine useless, it is safer to include an alternative way of calling `newQuery` to obtain $E(\mathcal{W}||n)$, contact the server, and feed the results to `newToken`. To complicate matters, we cannot assume that there is somebody to correctly transcribe the lengthy strings of the credit-refresh protocol.

We address this seemingly insurmountable problem by a bootable diskette containing a rescue program. The rescue program calls `newQuery` and writes the results to a file. The file is read by the software component running on a properly configured machine, the server is queried, and the results are written to a second file on the diskette. The contents of this second file are later fed to the hardware component. Note that like the software component, the rescue program is untrusted.

A.4 Forging tokens

The secure hash $H(n)$ is required in the response from the server to prevent an adversary from feeding forged tokens into stolen devices.

Let us assume that it is possible for an adversary to find out a Wanted-ID \mathcal{W}' that is not blacklisted. Sending $E(\mathcal{W}'||n')$ to the server, where n' is set to $n' = \mathcal{W}' \oplus (\mathcal{W} \oplus n)$, will result in a valid response, as \mathcal{W}' is not blacklisted. The value $\mathcal{W} \oplus n$ is taken from the server's prior response to a credit request from a stolen device with Wanted-ID \mathcal{W} , assuming the secure hash $H()$ is not used. By repeatedly intercepting these responses, which contain a null credit, and replacing them with the responses containing $\mathcal{W}' \oplus n'$ and a valid credit, the stolen device with Wanted-ID \mathcal{W} can be stopped from blocking.

Using in the server's responses a secure hash $H(n)$ instead of n , where $H(n)$ is by definition irreversible, prevents the adversary from forging tokens.

A.5 Feigning Theft

The ability to control a protected device through an independent server basically offers a level of indirection, hence increased flexibility.

Consider the case of an organization owning a large number of computing devices. A device is assigned to a member of the organization. The current common approach is to password-protect the computing devices. Typically, it is the assigned user who chooses the password. When a device is claimed to have been stolen, it is difficult for the organization to distinguish between theft by the assigned user (who is feigning theft) and theft by third-party. Indeed, the assigned user who happens to know the password can continue to use the device after the "theft." The same observation applies to Asset ID technology: the user can feign theft after having taken the device outside the protected premises.

The WANTED approach would prevent this attack. The organization would simply add the Wanted-ID of the device to the list of blacklisted devices. The device would run out of credit after a while and become useless. Similar reasoning could be applied to devices that often change hands, in particular rented devices.

Moreover, this system protects devices for which password protection is not applicable, for example, devices that lack a keyboard or screen. Lazy but otherwise honest users, who fail to set a password on the device assigned to them, are also protected. In particular, many users of mobile phones or personal digital assistants could benefit from WANTED.

B. Privacy

The PSN (Pentium Serial Number) present in all Pentium III chips has triggered strong negative reactions from the user community. Privacy has been considered a key issue already for decades [19].

At first glance, the Wanted-ID could be considered another "PSN." Indeed, if software had easy access to the Wanted-ID then nothing could prevent software from using it in a similar way as the "PSN." However, the hardware component never communicates the Wanted-ID to the software in the clear.

C. Scalability

On the protected device itself, the impact of the WANTED hardware and software components is minimal. The code executed in the hardware component has very few instructions and requires only a limited amount of memory. The most expensive instructions are a public-key encryption and a public-key signature verification. In terms of memory, only the credit counter, the grace period counter, the nonce, the Wanted-ID, and the public key of the server need to be stored. Moreover, the software component can be small because it is merely a relay between the WANTED hardware component and the server.

Given the large number of devices that could benefit from the WANTED system, scalability of the server is a key issue.

Clearly, not all WANTED-protected devices need to be assigned to the same server. The assignment can be based on administrative domains such corporations and ISPs. In this way, the processing load can be distributed.

The server does not need to keep a table of (Wanted-ID, *credit*) if the following improvement is made to the credit-refresh protocol presented earlier: A default credit value c_0 is added to the return value of `newQuery`, which becomes $E(\mathcal{W}||n||c_0)$. The server can then assign $c := c_0$ in its response. The default credit value is set by the legitimate owner of the device after proper authentication. The default credit value must be stored securely on the hardware component.

To obtain a different amount of credit, say $c_1 \neq c_0$, the legitimate user asks the server to add its Wanted-ID to a customized-credit list with credit amount c_1 . If a Wanted-ID is on the customized-credit list, the server overrides c_0 with c_1 in its answer to the client's query. To summarize, the server needs to keep

1. a blacklist containing the Wanted-IDs of stolen devices, and
2. a customized-credit list containing a list of Wanted-IDs with their associated customized credit.

In the normal case, i.e. the device is neither stolen nor does it require a customized credit, its Wanted-ID will not appear on either of the two lists. Thus, in practice both lists will be rather small. This should further help to maintain the WANTED system scalable.

IV. FUTURE WORK

Here we have focused on specific elements of the WANTED system. Future work will address issues such as: How are server public keys distributed? How is the server's public key put on the hardware component? What type of credentials are needed for this operation? What happens when protected devices are resold? Who is allowed to deliver proofs of property?

While a complete theft-deterrent system needs to integrate many more aspects than those presented here, the specific piece we presented has the advantage of being relatively stand-alone. In a different context, secure electronic payment protocols [20, 21] have to consider similar aspects to allow their wide deployment.

V. CONCLUSION

The Internet has become the favorite communication platform for users worldwide. The next step may well be that all sorts of devices become interconnected, and interact without user intervention.

In the world of pervasive computing, an entire new set of problems needs to be addressed, such as ad-hoc networking and auto-configuration. In this context, theft deterrents have to be carefully considered, although only very few contributions exist in this field. We have shown a method for dealing with a large number of devices through a simple credit-based mechanism resistant to various types of attacks. At the same time, important issues such as privacy and scalability have been considered. Our mechanism has low hardware requirements, which are met by most cryptography hardware.

REFERENCES

- [1] IBM, "Asset ID," <http://www.pc.ibm.com/ww/assetid/index.html>.
- [2] Absolute Software Corp., "Computrace," <http://www.computrace.com/>.
- [3] Absolute Software Corp., "Security apparatus and method," Patent WO 9615485, May 1996.
- [4] Computer Sentry Software Inc., "Cyberangel," <http://www.sentryinc.com>.
- [5] P.D. Collins, K.W. Royer, and M.D.J. Bowyer, "Immobilisation protection system for electronic components," Patent WO 9894967, February 1998.
- [6] G.J. Proudler, "Security device," Patent EP 0740037, October 1996.
- [7] A. Abdul-Rahman and S. Hailes, "A distributed trust model," in *Proc. New Security Paradigms Workshop (NSPW-97)*, New York: ACM, 1997, pp. 48-60.
- [8] A. Josang, "A trust policy framework," in *ICIS: Int'l Conf. on Information and Communications Security, Lecture Notes in Computer Science*, Berlin Heidelberg: Springer, 1997.
- [9] R.J. Anderson and M.G. Kuhn, "Tamper resistance – a cautionary note," in *The Second USENIX Workshop on Electronic Commerce Proc.*, Oakland, CA, Nov. 1996, pp. 1-11.
- [10] R.J. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices," in *Lecture Notes in Computer Science*, vol. 1361. Berlin Heidelberg: Springer, 1998, p. 125.
- [11] D.R. Stinson, *Cryptography - Theory and Practice*, Boca Raton, FL: CRC Press, 1996.
- [12] R. Rivest, *RFC 1321: The MD5 Message-Digest Algorithm*, IETF, April 1992.
- [13] J. Kelsey, B. Schneier, and D. Wagner, "Protocol interactions and the chosen protocol attack," in *Lecture Notes in Computer Science*, vol. 1361. Berlin Heidelberg: Springer, 1998, pp. 91-104.
- [14] C. Meadows, "Formal verification of cryptographic protocols: A survey," in *Proc. Asiacrypt 96*, 1996.
- [15] K.J. Compton and S. Dexter, "Proof techniques for cryptographic protocols," in *Lecture Notes in Computer Science*, vol. 1644. Berlin Heidelberg: Springer, 1999, p. 25.
- [16] M. Bellare and P. Rogaway, "The exact security of digital signatures — how to sign with RSA and Rabin," in *Lecture Notes in Computer Science*, vol. 1070. Berlin Heidelberg: Springer, 1996, p. 399.
- [17] W. Milliken C. Partridge, and T. Mendez, *RFC 1546: Host anycasting service*, IETF, November 1993.
- [18] D. Aucsmith, "Tamper resistant software," in *Lecture Notes in Computer Science*, vol. 1174. Berlin Heidelberg: Springer, 1996, p. 317.
- [19] A.R. Miller, *The Assault on Privacy: Computers, Data Banks and Dossiers*, New York: Mentor Books, 1971.
- [20] M. Bellare, J. A. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner, "iKP—a family of secure electronic payment protocols," in *Proc. First USENIX Workshop on Electronic Commerce*. Berkeley: USENIX Assoc., 1995, pp. 89-106.
- [21] J.-P. Boly, A. Bosselaers, R. Cramer, R. Michelsen, S. Mjølsnes, F. Muller, T. Pedersen, B. Pfitzmann, P. de Rooij, B. Schoenmakers, M. Schunter, L. Vallée, and M. Waidner, "The ESPRIT project CAFE – high security digital payment systems," in *Proc. Third European Symp. on Research in Computer Security (ESORICS)*, D. Gollmann, Ed. Brighton, UK, Nov. 1994, *Lecture Notes in Computer Science*, vol. 875. Berlin Heidelberg: Springer, 1994.