Protection of Data Groups from Personal Identity Documents

Przemysław Kubiak, Mirosław Kutyłowski and Wojciech Wodo

***Abstract*—For personal identity documents, we propose a procedure of presenting a signed face image of the document holder. Our goal is to authenticate the image by document issuer, but at the same time to prevent misuse of this high quality digital data. As the signature is recipient dependent, illegitimate transfer of the signature to third parties is strongly discouraged. Despite that the document issuer is the signatory and that the image recipients are unpredictable in advance, only a very limited amount of information has to be stored on a chip of the personal identity document. Moreover, the solution prevents creating additional signatures by document issuer, as a signature created outside the card leads to a mathematically strong proof of a fraud.**

Although motivation for the protocols presented below was protection of biometric data, the protocols might be used in case of any data.

***Index Terms*—personal identity document, smart card, personal data protection, designated recipient, electronic signature, Merkle tree**

I. Problem Description

A. Background problem

A personal identity document equipped with a cryptographic chip, called *e-ID* for short, offers high level security guarantees against document forgeries: while there is a race between graphical protection techniques and the forgery methods. On the other hand, repeating the same data in electronic form and signing them by the document issuer provide strong and independent security mechanisms at a low price. Advances in cryptanalysis limit the long-term value of these guarantees, nevertheless they are relatively long-lasting.

Electronic layer of e-ID may store a high resolution face image of the document holder – more detailed than the image printed on the document. This enables much more reliable inspection based on e-ID. The strategy applied in particular by biometric passports is to present not only raw data, but also a signature of the document issuer for those data. In this way during an inspection we may become convinced that the image presented originates from the document issuer and has not been replaced even if chip security of e-ID has been broken.

Securing data with a signature of the document issuer is a two-edged sword. Once the signature is created, it can be used by anybody to confirm authenticity of digital data. Therefore this approach leads to privacy threats: once the signed data is shown to a second party, the owner of e-ID has no further control over who has access to it. In particular, this data can be sold to third parties. The signature has a negative influence on the situation, as quality of the data is confirmed by authority issuing the e-ID. This problem has been one of the major factors behind the design of German personal identity card, where the data might be shown without issuer's signature, but via an authenticated and secure channel [1]. The communication and authentication protocols are designed in such a way that even a full transcript of a session together with ephemeral data created during the session on the terminal side cannot be used as a proof against a third party. This is achieved by means of *simultability*. The price is that we have to assume that the chips of the personal identity cards provide full security against all kinds of (practical) attacks.

Authors are with Wroclaw Univeristy of Technology, Institute of Mathematics and Computer Science, e-mail: *firstname.lastname*@pwr.wroc.pl

This research was initiated under support of Foundation for Polish Science.

B. Assumptions about e-ID chips.

We assume that the chip used by e-ID provides certain (limited) security against the issuer of e-ID. Namely, we assume that keys generated privately on the chip can be read by the e-ID issuer as long as the key generation process takes place in environment controlled by the issuer. However, keys generated on the chip when the e-ID is in control of the owner are neither predictable for the e-ID issuer nor they leak from the e-ID.

The assumptions above reflects the setting where the chip vendor does not collude with the authority issuing and personalizing identity documents, but the authority has access to technologies that may break security means on the chip and can access all relevant data on the chip.

C. System goals.

We aim to provide a solution such that:

- Once the face image (or more generally, the data groups containing personal data of the owner) are presented by an e-ID, then a customized signature of the document issuer is attached.
- The signature indicates the recipient of the signature, but the proof is not necessarily unconditional. This means, it should provide traces who is not fulfilling the duties of personal data protection, but on the other hand the signature is not necessarily an undeniable proof of e-ID document presence.
- The authority issuing the e-ID documents cannot create clone documents and customized signatures in order to accuse a certain party for violations of personal data protection.

The simplest solution is to provide a signature $\mathrm{Sign}_K(H(D), R)$, where K is the signing key of the

issuing authority, D denotes the data groups and R is the recipients ID. There are two severe problems with this approach: R must be known in advance and the issuer can create these signatures at any time, distribute them and accuse R of violations of personal data protection.

The first problem can be dealt with by means of proxy signatures [2]: the chip of e-ID receives data that enable it to create signatures on behalf of the document issuer. However, with this approach we solve one hard problem, but create a new harder one. Namely, once an adversary breaks into a chip of e-ID, it can manipulate the e-ID document and in particular replace the face image.

One may also try to use designated verifier schemes - in this case the signature is worthless to anybody, but the verifier determined at signature creation time. The same problems apply as before – the issuing authority has either to create them in advance and store on the chip of e-ID or use a proxy version of it. Moreover, proxy and designated verifier signature schemes are significantly more complicated than the standard signatures, use operations that might be unavailable on the standard chips. Therefore the non-volatile memory requirements for storing program code and data might be quite high regarding limitations for chips on smart cards. Finally, there is nothing so far that would prevent malicious authorities from creating and using the clones of identity documents.

Another option is hiding the signature of the issuing authority by the e-ID. Instead, the chip proves that it holds a signature for given data D (compare [3]). However, such solutions fall into another category as the verifier cannot store the signature for offline verification. Our goal is a real signature - the only difference should be that it has to be customized to show the original recipient.

D. Our contribution

We present two solutions with slightly different properties. The first one is based on hash functions, the second one on asymmetric techniques. In both cases the signature is customized in a way that points to the signature recipient and it is infeasible to change this pointer unless one has access to the secrets stored in the chip of e-ID document.

II. Hash Based Protocol

Below we sketch the idea of our solution.

A. General settings.

The document issuer holds a conventional pair of keys for creating electronic signatures. Authentication of the public key is achieved again in the standard way (e.g., by publishing or by public key certificates).

For each e-ID document we have k different positions for document verifiers, each verifier is assigned one position. The number k is a system parameter and its value has to be fine tuned depending on system size and trade-off between privacy and detectability of parties misusing personal data. The position of a verifier for each e-ID is determined separately in a pseudorandom way. Namely, for a hash function H, a verifier V for identity document ID_D is assigned position $H(V, D) \bmod k$. In this way, for a given identity document there are good chances that the verifiers the owner of the document visits most frequently have been assigned different positions. On the other hand, without knowledge of the datagroups the position of a given verifier in a given e-ID is completely unpredictable.

As the positions will correspond to leaves of a Merkle tree constructed separately for each e-ID, we assume that k is a power of 2 and throughout the paper log symbol defines the binary logarithm.

B. Document personalization by the e-ID issuer.

For each e-ID document ID_D there is a master secret S_D chosen uniformly at random by the document issuer.

According to standard conventions, we assume that data stored on ID_D consist of data groups D=(D_1,...,D_m), where each D_i, $i = 1, \ldots, k$, is a single data group. As the data might be exposed selectively, the signature is created for $H_D = H(H(D_1), \ldots, H(D_m))$. In this way, for verification of a signature it suffices to present $H(D_1), \ldots, H(D_m)$ as well as the data groups D_j that are to be disclosed.

For the purpose of clone-evidence we need secrets x_D - e.g., x_D might be a signature of the document issuer under the text *"ID_D has been cloned or broken"*.

For ID_D the document issuer creates a Merkle tree [4] of height $\log k + 2$ in the following way:

- for each $i < k$ there are 4 corresponding leaves; they are labelled with the following values: H_D, $x_{D,R,i}$, H_D, $x_{D,L,i}$, where $x_{D,L,i} = R(i||S_D)$, $x_{D,R,i} = x_D - x_{D,L,i}$, and R is a cryptographic pseudorandom generator.
- We construct the labels for higher levels of the tree as always for Merkle trees: if a node A has children nodes with labels h_1 and h_2, then the label of A is $H(h_1, h_2)$.

Let $Root_D$ denote the label of the root of the tree constructed for ID_D. The last step is to create a signature $Sign_D$ of $Root_D$ by the document issuer and to store it on the chip of ID_D.

C. E-ID personalization by the owner.

After delivering ID_D to its owner, it executes a procedure of uploading a random secret X_D to the chip of ID_D. X_D can be kept outside ID_D, but must be unknown for the document issuer.

The purpose of X_D is to determine the leaves used for signing: if $H(i, X_D) \bmod 2 = 0$, then for position i the leaf labeled with $x_{D,L,i}$ is used. If $H(i, X_D) \bmod 2 = 1$, then for position i the leaf labeled with $x_{D,R,i}$ is used. Here we assume that hash function H is cryptographically secure, thus there is no bias for any single bit position.

In particular, the values of $H(i, X_D) \bmod 2$ may be stored in an array A of k bits.

D. Customizing the signature.

Assume that a verifier V is to receive signed data from document ID_D. Apart from $H(D_1), \ldots, H(D_m)$, and chosen data groups D_j the e-ID prepares a signature of the document issuer in the following way:

- compute position i for V as $i = H(V, D) \bmod k$,
- determine a *path* P_i from a leaf holding $x_{D,Z,i}$ to the root, where $Z = L$ if $H(i, X_D) \bmod 2 = 0$, and $Z = R$ otherwise,
- compute a list HP_i of hashes: for each node of P_i, the list HP_i indicates the label of the sibling of the node on P_i. The only exception is the leaf node, for which its label is given and not the label H_D of the sibling node.
- return $H(D_1), \ldots, H(D_m)$, HP_i, $Sign_D$ and the relevant data groups D_j which are to be disclosed.

E. Verification of a signature.

The following steps are necessary to verify a signature $H(D_1), \ldots, H(D_m)$, HP_i, $Sign_D$:

- $H(D_1), \ldots, H(D_m)$ are checked against the data groups disclosed to the verifier,
- the hash values on the path P_i are reconstructed using HP_i, the first value is computed as $H_D = H(H(D_1), \ldots, H(D_m))$,
- the signature $Sign_D$ is verified in the conventional way, against the label $Root_D$ of the root node computed in the previous step.

F. Implementation issues - speeding up signature creation.

Note that the chip does not need to remember the labels of nodes of its hash tree – it can be reconstructed from D_1, $\ldots D_m$ and the secret S_D. Also it is easy to see that auxiliary storage required to compute HP_i is roughly $\log k$ hash values.

If k is relatively small, then computation effort on the chip is acceptable. However, if this is not the case, we can significantly reduce the computational effort by storing the labels of the nodes at height $\frac{1}{2}\log k + 1$ of the tree. In this case the chip has to reconstruct labels for *two* subtrees of depth $\frac{1}{2}\log k + 1$ of total size roughly $6\sqrt{k}$ instead of $\approx 6k$.

G. Clone detection.

As the secret X_D is created *after* the e-ID document is given to the owner, the issuer cannot guess which leaves are used by the chip of e-ID for each position i. A single attempt to create an extra signature on behalf of the document owner leads with probability $\frac{1}{2}$ to disclosure of the secret x_D. An attempt to create, say 20, such signatures will not lead to fraud disclosure with probability $\frac{1}{2^{20}}$, which is the value too low for any authority to dare a fraud.

H. Detection of offenders of personal data protection.

Assume that a verifier V collects data and signatures obtained from e-ID documents. Assume that V has sold n such records to a data bank L which has reached the total size N. Assume that L has been captured by law enforcement authorities.

For each signature found in L we can check if it is possible that it has been obtained from V. If a signature uses the same position in the Merkle tree as it would be used for V, then we say that this is an *accusation* against V. As the positions in the Merkle tree are determined in a pseudorandom way, we may assume that the expected number of accusations against V in L equals

$$(N - n) \cdot \tfrac{1}{k} + n = \tfrac{N}{k} + n(1 - \tfrac{1}{k}).$$

If V is honest, then the expected value equals $\frac{N}{k}$.

Statistical tests indicating dishonest behavior of V can be based on the fact that the Bernoulli distribution is fairly concentrated. For instance, according to Chernoff bounds, probability that there are more than $\frac{2N}{k}$ accusations in case of honest V is bounded by $(e/4)^{N/k}$. For $k = 16$ and $N = 2^{10}$ we get that probability to get more than 128 accusations is $\approx 2^{-35}$, while the expected number of accusations for dishonest V and 70 records sold is higher than 129. This shows that any large scale sale of data is very risky for a verifier. On the other hand, in this kind of business what counts is only large scale sale, as single records have a low price.

Note that higher values of k make detection of dishonest verifiers more reliable. On the other hand, if k is low, then a signature pointing to position i which should be used by V is not an evidence that ID_D has been presented to V. Namely, this position is used by the fraction $\frac{1}{k}$ of all verifiers!

I. Feasibility Issues

We have performed speed tests on Gemalto Java Cards concerning computation of hash values. The results for exemplary parameters are as follows:

SHA-1 (160 bits): 1 hash $\approx$ 5ms, 1280 hashes $\approx$ 4.8s,
SHA-2 (256 bits): 1 hash $\approx$ 9ms, 1280 hashes $\approx$ 10s.

For comparison observe the number of hashes to be computed to create a single signature for tree depth 10 when the hashes of level 5 (32 values) are stored by the chip, is $2 \cdot (32 - 1)$, so the time required is less than 0.5s for SHA-2.

Memory usage for data in case of trees of depth 10 (with intermediate level at depth 5 stored on the chip) equals:

keys:	master secret S_D – 128 bits, user secret X_D – 128 bits, array of hash values on Merkle tree on intermediate level at depth 5 – $32 \cdot 256 = 8192$ bits)
temp. hash values:	at most 6 hashes at a time – 1536 bits.

III. Asymmetric Approach

In this section we sketch a protocol which can be used to create customized signatures by *tagging* a signature of the document issuer. Namely, the chip of e-ID attaches a tag to the data groups and the signature of the issuing authority revealed to a verifier. The point is that without the tag signature verification is infeasible, and that the tag indicates the intended verifier. No prior agreement on the identity of verifiers is necessary.

A. Building Blocks

The main building block for the high-resolution protocol is a solution used to prove equality of two discrete logarithms.

a) System settings.: Let g generate a group of prime order q. Furthermore, assume that Decisional Diffie-Hellman Problem is hard for this group. Let h belong to this group be chosen so that its discrete logarithm is unknown.

We assume that a prover holds a private exponent x. The goal of the prover is to convince that two elements a, b have the form $a = g^x$, $b = h^x$.

b) Schnorr-like proof of equality of discrete logarithms [5].: First the prover performs the following steps:

1) generate r at random,
2) $k := g^r$, $\ell := h^r$,
3) $e := H(k, \ell, g, h, a, b, m)$, where m is some message, for example an empty message or *the name of the addressee of the proof*, i.e. the name of the intended verifier,
4) $s := r + ex \bmod q$,
5) send (e, s) to the verifier.

Then the verifier performs the following steps:

1) $k' := g^s/a^e$,
2) $\ell' := h^s/b^e$,
3) $e' := H(k', \ell', g, h, a, b, m)$,
4) return $\texttt{ok}$ if $e = e'$.

B. Sketch of the Scheme

The system is supported by a card management system called below CAMS. We refer also to standard protocols for chip authentication (Chip Authentication or ChA) and authenticating terminals (Terminal Authentication or TA) [1].

1) Document personalization.: For each single identity document the following steps are executed by issuing authority:

1) All but two data groups for the e-ID are completed in advance, and are stored in some registry on the side of CAMS.
2) The data groups are copied to the chip of e-ID.
3) The private key and the corresponding public key for ChA are generated by the e-ID chip.
4) The ChA public key is copied to the data groups (i.e., to a copy stored locally on the e-ID chip as well to a copy stored in the registry of CAMS).

The data groups are still not authenticated by the issuing authority. The e-ID is in a state we call "red", which means that all functions of the chip are blocked – only Terminal Authentication and Chip Authentication with terminals of CAMS are allowed.

When the e-ID is in hands of its owner, it must be unblocked. In a private environment the owner connects to a service of CAMS and after mutual authentication via TA and ChA protocols the following steps are executed:

1) The e-ID chip generates its private key $\tilde{x}$ for tagging, and computes $\tilde{a} = g^{\tilde{x}}$, where g is fixed for all users.
2) Key $\tilde{a}$ is written in the remaining empty data group, both in the e-ID chip and in its record in the CAMS registry.
3) The e-ID chip and CAMS each compute $\tilde{h} = H_g(D)$, where H_g is a hash function with the image included in the group generated by g.
4) The e-ID chip computes $\tilde{b} = \tilde{h}^{\tilde{x}}$ and sends $\tilde{b}$ to CAMS.
5) The e-ID chip and CAMS execute zero-knowledge protocol for equality of discrete logarithms for $\tilde{a}, \tilde{b}$ and the corresponding bases $g, \tilde{h}$ (here Schnorr-like protocol described above has to be used, m is chosen to be the string "CAMS").
6) The e-ID chip enters a "yellow" state, which is intermediate between the red one and the "green" one for regular usage. The e-ID chip disconnects from CAMS.

The next phase is generating signature of the issuing authority:

1) User's data groups from CAMS's registry are transferred together with the proof of equality of discrete logarithms to the document issuing authority.
2) The document issuing authority verifiers the proof and if the verification result is positive, then it creates a signature $Sign(\tilde{b})$ under $\tilde{b}$.
3) $Sign(\tilde{b})$ is transferred back to CAMS's registry.

If an e-ID is in the "yellow" state, then any time the e-ID is used it tells the middle-ware to connect to CAMS's service to fetch $Sign(\tilde{b})$. If the signature is available, it is transferred to the chip of e-ID through a secure channel (established by means of TA and ChA protocols). The e-ID verifies the signature, if it is correct, then the e-ID switches from the "yellow" state to the "green" one.

2) Data Group Authentication: To execute this part the e-ID must be in "green" state. After completion of the terminal authentication and the chip authentication procedures the terminal of the verifier and the e-ID chip execute the following protocol (we assume that the terminal is allowed to obtain the whole data D):

1) The e-ID chip sends D and $Sign(\tilde{b})$ to the terminal.
2) The terminal reads $\tilde{a}$ from D and computes $\tilde{h} = H_g(D)$.
3) The e-ID chip computes $\tilde{h} = H_g(D)$ and $\tilde{b} = \tilde{h}^{\tilde{x}}$ and sends $\tilde{b}$ to the terminal (now both sides know the tuple $(\tilde{a}, \tilde{b}, g, \tilde{h})$ and $Sign(\tilde{b})$, but the link between $\tilde{h}$ and $\tilde{b}$ must be proven by the e-ID chip).
4) Both parties execute equality of discrete logarithms protocol for $\tilde{a}, \tilde{b}$ and the corresponding bases $g, \tilde{h}$. Schnorr-like protocol is used for m being a string identifying the verifier.

C. Discussion

As in case of the protocol from Section II the issuing authority cannot create a clone of an e-ID document without breaking into the e-ID chip and reading the secrets installed there by the owner of the document.

Unlike in the previous solution, we are free to make tags as precise as we want: the message m included in the proof of equality of discrete logarithms may fully indicate the verifier's identity. On the other hand, it is also possible to insert restricted information only – as for the protocol from Section II. In the former case the tags are undeniable proofs

that an e-ID has issued a customized signature for the verifier indicated in the tag.

Apart from tagging, an e-ID document may check the rights of the terminal to get the data. This can be achieved in a standard way where the terminals are authenticated by certificates and the underlying PKI infrastructure (compare [6]).

Finally, let us remark that despite cryptographic countermeasures and legal restrictions, any party can sell a set of *unauthenticated* personal data. Authentication may be statistical – the party buying the set may confront it with the set of locally stored data. If the records belonging to the intersection set are the same, then the whole set bought is assumed to be correct[1]. In order to prevent such situation, the e-ID could insert steganographic data in images revealed to the verifiers (with such steganographic tags the data would depend from the intended addressee). However, it is a hard challenge to design such protocols: apart from all problems known so far for steganographic security measures we have to deal with the problem of low computational resources on the e-ID chip.

Another option to limit illegal selling of personal data, which may always undergo statistical verification, is to require by law that each record containing personal data should be associated with

- a tag proving that the party that stores the record has obtained it directly from the smart card,
- or a consent signed by the person for selling/revealing her/his data,
- or a pointer to some legal regulations that imposes a duty on the party to process the data (however, the data should still be associated with the tags, indicating whom the data were initially revealed by smart cards).

Then in case of an audit a party that stores the data is safe.

Moreover, each party that sees personal data with the tag issued for another party, and without consent of the citizen for selling/revealing her/his data, should be obliged by law to inform the authorities about the leak (the data seen should be attached to the information). In cases when a party is legally binded to reveal the data to another party it should obtain a signed request for the data, to avoid being accused for data leakage.

IV. Security of the Asymmetric Approach

A. Problem Statement

The exponentiation $\tilde{h}^{\tilde{x}}$, where $\tilde{h} = H_g(D)$, used in the protocol from Section III resembles BLS signature scheme [7]. However, if $\langle g \rangle$ would be a pairing friendly group, no ZKP-EDLP (Zero-Knowledge Proof of Equality of Discrete Logarithms) would be necessary, because equality could immediately be checked with pairing.

Thus augmenting the exponentiation with ZKP-EDLP we obtain an analog of BLS signature scheme in pairing unfriendly groups. Since D is of the form $(g^{\tilde{x}}, M)$, where M are some data, we obtain a kind of a self-signed certificate of the public key $\tilde{a} = g^{\tilde{x}}$. The document issuing authority makes signature $Sign(\tilde{b})$ under the BLS-like "signature" value $\tilde{b} = \tilde{h}^{\tilde{x}}$.

Problem: *is it feasible to change M and tune $\tilde{x}$ accordingly in such a way that $\tilde{b}$ remains unchanged?* The protocol from Section III assumes negative answer to this question.

B. Argument for Security

We have Schnorr-like dependency here: some randomizer is used inside and outside the hash function: $\tilde{b} = (H_g(g^{\tilde{x}}, M))^{\tilde{x}}$. Hence when we try to change M to M' we search for $x' \in \mathbb{Z}_q^*$ yielding a collision:

$$\tilde{b}^{(x')^{-1}} = H_g(g^{x'}, M').$$

Probability of such an event is not greater than probability of the following collision

$$\tilde{b}^{(x')^{-1}} = H_g(y, M'),$$

where x', y could be independently chosen. But the latter collision occurs no more frequently than the collision

$$\tilde{b}^{(x')^{-1}} = H_g(\tilde{M}), \tag{1}$$

where $\tilde{M}$ could be any bitstring. In the random oracle model for H_g probability of the last event results from the birthday paradox in two rooms setting: Let fix parameter $\gamma \in (0, 1)$. Provided that in each single choice of $(x', \tilde{M})$ an element $\tilde{b}^{(x')^{-1}} \in \mathrm{Im}(H_g)$, the number of choices $(x', \tilde{M})$ yelding collision (1) with probability no smaller than γ is equal to $c_\gamma \cdot \sqrt{|\mathrm{Im}(H_g)|}$, where constant c_γ results from the birthday paradox mentioned above, and is dependent of γ. Since $x', \tilde{M}$ could be chosen independently, the expected number of choices of $(x', \tilde{M})$ to obtain a collision (1) with probability no smaller than γ, equals in the random oracle model for H_g to

$$\frac{c_\gamma \cdot \sqrt{|\mathrm{Im}(H_g)|}}{\Pr\left(\tilde{b}^{(x')^{-1}} \in \mathrm{Im}(H_g)\right)}.$$

V. Conclusions

It turns out that protection of high quality personal data disclosed by personal identity cards is feasible in the model in which there are trust limitations against smart cards manufacturers and authorities issuing the identity documents. Moreover, standard smart cards with cryptographic functions can be used for implementing such a solution.

[1] See that if the issuing authority creates a duplicate of a document with the same personal data but with different key material, then it could be detected by parties already storing data from the original document. Of course a list of revoked chips should be available online to prevent misuse of cards stolen or lost.

References

[1] BSI, "Advanced Security Mechanisms for Machine Readable Travel Documents 2.10," Technische Richtlinie TR-03110, 2010.

[2] M. L. Das, A. Saxena, and D. B. Phatak, "Algorithms and approaches of proxy signature: A survey," *I. J. Network Security*, vol. 9, no. 3, pp. 264–284, 2009.

[3] J. Monnerat, S. Pasini, and S. Vaudenay, "Efficient deniable authentication for signatures," in *ACNS*, ser. Lecture Notes in Computer Science, M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, Eds., vol. 5536. Springer, 2009, pp. 272–291.

[4] R. C. Merkle, "Secrecy, authentication, and public key systems." Ph.D. dissertation, Stanford University, Stanford, CA, USA, 1979, aAI8001972.

[5] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *CRYPTO*, ser. Lecture Notes in Computer Science, E. F. Brickell, Ed., vol. 740. Springer, 1992, pp. 89–105.
[6] BSI, "PKIs for Machine Readable Travel Documents 1.10," Technische Richtlinie BSI TR-03129, 2009.
[7] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

Przemysław Kubiak Przemysław Kubiak received master degree in mathematics in 1997 and Phd in 2001. He is an assistant professor for computer science at Wrocław University of Technology.

His research interests are in efficient algorithms for public-key cryptography, and in reducing the need for trust in components of cryptographic protocols.

Mirosław Kutyłowski

Miroslaw Kutylowski is a full professor of computer science at Wroclaw University of Technology, member of Scientific Council of Institute of Computer Science, Polish Academy of Sciences. He received PhD and Habilitation degree from Wroclaw University. Former Alexander von Humboldt-Fellow at Darmstadt Technical University, Hochschuldozent at Paderborn University. He received Mistrz Prize from Foundation for Polish Science in 2009 and IBM Faculty Award in 2012. Member of Steering Committee of ESORICS.

Prof. Kutylowski specializes in ad hoc systems, privacy and security, including in particular protocols for personal identity cards.

Wojciech Wodo

Wojciech Wodo is a PhD candidate of computer science at Wroclaw University of Technology, graduate of special "Top 500 Innovators" program at UC Berkeley focused on science management, technology transfer, commercialization and university-industry collaboration. He received MSc degree from Wroclaw University of Technology in computer science. Mr. Wodo worked as a technology transfer specialist in Wroclaw Research Center EIT+ (2011-2012).

He is a vice-president of MANUS Foundation , responsible for academic entrepreneurship development and consultancy for start-ups.

His field of interest are identifications and authentications methods based on biometric factors as well as cryptographic protocols.