Ubic: Bridging the Gap between Digital Cryptography and the Physical World

Mark Simkin[1], Dominique Schröder[1], Andreas Bulling[2], and Mario Fritz[2]

[1] Saarland University
Saarbrücken, Germany
[2] Max Planck Institute for Informatics
Saarbrücken, Germany

Abstract. Advances in computing technology increasingly blur the boundary between the digital domain and the physical world. Although the research community has developed a large number of cryptographic primitives and has demonstrated their usability in all-digital communication, many of them have not yet made their way into the real world due to usability aspects. We aim to make another step towards a tighter integration of digital cryptography into real world interactions. We describe Ubic, a framework that allows users to bridge the gap between digital cryptography and the physical world. Ubic relies on head-mounted displays, like Google Glass, resource-friendly computer vision techniques as well as mathematically sound cryptographic primitives to provide users with better security and privacy guarantees. The framework covers key cryptographic primitives, such as secure identification, document verification using a novel secure physical document format, as well as content hiding. To make a contribution of practical value, we focused on making Ubic as simple, easily deployable, and user friendly as possible.

Keywords: Usable security, head-mounted displays, ubiquitous cryptography, authentication, content verification, content hiding.

1 Introduction

Over the past years, the research community has developed a large number of cryptographic primitives and has shown their utility in all-digital communication. Primitives like signatures, encryption schemes, and authentication protocols have become commonplace nowadays and provide mathematically proven security and privacy guarantees. In the physical world, however, we largely refrain from using these primitives due to usability reasons. Instead, we rely on their physical counterparts, such as hand-written signatures, which do not provide the same level of security and privacy. Consider the following examples:

Authentication. In practice, most systems, such as ATMs or entrance doors, rely on the two-factor authentication paradigm, where a user, who wants to authenticate himself, needs to provide a possession and a knowledge factor. At an ATM, for instance, the user needs to enter his bank card and a PIN in order to gain access to his bank account. Practice has shown that this type of authentication is vulnerable to various attacks [1,2,3], such as skimming, where the attacker mounts a little camera that films the PIN pad and a fake card reader on top of the actual card reader that copies the card's content. Here, the fact that users authenticate with fixed credentials is exploited to mount large scale attacks by attacking the ATMs rather than specific users.

Hand-written signatures. Physical documents with hand-written signatures are the most common form of making an agreement between two or more parties legally binding. In contrast to digital signatures, hand-written signatures do not provide any mathematically founded unforgeablility guarantees. Furthermore, there is no well-defined process of verifying a hand-written signature. This would require external professional help, which is expensive, time consuming, and therefore not practical.

Data privacy. Todays workplace is often not bound to specific offices or buildings any more. Mobile computing devices allow employees to work from hotels, trains, airports, and other public places. Even inside office buildings, novel working practices such as 'hot-desking' [4] and 'bring your own device' [5] are employed more and more to increase the employee's satisfaction, productivity, and mobility. However, these new working practices also introduce new privacy threats. In a mobile working environment, potentially sensitive data might be leaked to unauthorized individuals, who can see the screen of the device the employee is working on. A recent survey [6] of IT professionals shows that this form of information theft, known as *shoulder surfing*, constantly gains importance. 85% of those surveyed admitted that they have at least once seen sensitive information that they were not supposed to see on somebody else's screen in a public place. 80% admitted that it might be possible that they have leaked sensitive information at a public place.

In this work, we present Ubic, a framework and prototype implementation of a system that allows users to bridge the gap between digital cryptography and the physical world for a wide range of real world applications. Ubic relies on *head-mounted displays* (HMDs), like Google Glass[1], resource-friendly computer vision techniques as well as mathematically sound cryptographic primitives to provide users with better security and privacy guarantees in all of the scenarios described above in a user-friendly way. Google Glass consists of a little screen mounted in front of the user's eye and a front-facing camera that films the users view. It supports the user in an unobtrusive fashion by superimposing information on top of the users view when needed.

1.1 Contributions

To make a contribution of practical importance, in this work we focus on providing a resource-friendly, easy-to-use system, that can be seamlessly integrated into the current infrastructures. Ubic offers the following key functionalities:

[1] `https://www.google.com/glass/`

Authentication. We use a HMD in combination with challenge-response protocols to allow users to authenticate themselves in front of a device, such as an ATM or a locked entrance door. In contrast to current solutions, the PIN is not fixed but generated randomly each time. Neither does an attacker gain any information from observing an authentication process, nor does he gain any from compromising the ATM or the bank, since they can only generate challenges, but not solve them. Copying the card does not help the attacker, since it does not contain any secret information, but merely a public identifier.

Content Verification. We enable the generation and verification of physical contracts with mathematically proven unforgeability guarantees. For this purpose, we propose a new document format, VeriDoc, that allows for robust document tracking and optical character recognition, and contains a digital signature of its content. Using the HMD, a user can conveniently and reliably verify the validity of the document's content.

Two-Step Verification. Based on the signature functionality described above, we introduce *two-step verification* of content. During an online banking session, for instance, a user might request his current account balance. This balance is then returned along with a signature thereof. Using the HMD, we can verify the signature, and therefore verify the returned account balance. In this scenario, an attacker would need to corrupt the machine that is used for the banking session and the HMD at the same time in order to successfully convince the user of a false statement.

Content Hiding. We provide a solution for ensuring privacy in the mobile workplace setting. Rather than printing documents in plain, we print them in an encrypted format. Using the HMD, the user is able to decrypt the part of the encrypted document that he is currently looking at. An unauthorized individual is not able to read or decrypt the document without the corresponding secret key. Companies commonly allow employees with certain security clearances to read certain documents. We use predicate encryption to encrypt documents in such a way that only employees with the requested security clearances can read them.

1.2 Smartphones vs. Head-Mounted Displays

It might seem that all of the above scenarios could also be realized with a smartphone. This is not the case. In the content hiding scenario, we rely on the fact that the decrypted information is displayed to the user right in front of his eye. A smartphone is still vulnerable to shoulder-surfing and would therefore not provide any additional privacy guarantees. Realizing the authentication scenario with a smartphone is also problematic because a loss of possession is hard to detect and therefore an attacker might gain access to all secret keys as soon as he obtains the phone. Requiring the user to unlock the phone before each authentication process does not solve the problem, since the attacker might simply observe the secret that is used to unlock the phone.

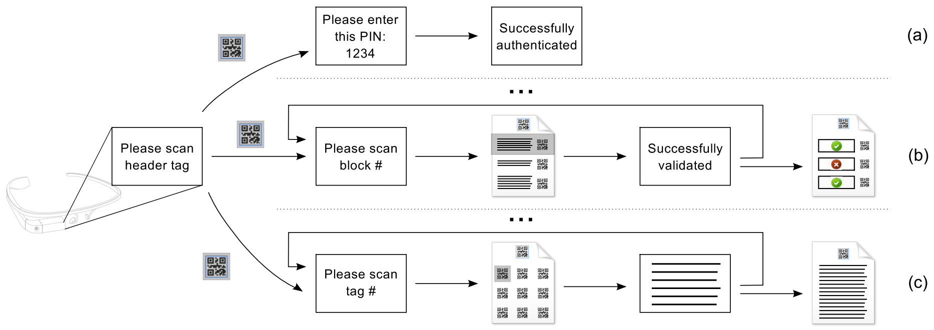

Fig. 1. Overview of the Ubic processing and interaction pipeline for the different operation modes: identification (a), content verification (b), and content hiding (c). The user starts each interaction by scanning the header 2D-barcode (indicated in gray). Ubic then guides the user through each process by providing constant visual feedback.

Ubic overcomes this problem by using the so-called *on-head detection* feature of the Google Glass device[2]. The device is notified whenever it is taken off and at this point, Ubic removes all keys from the memory and only stores them in an encrypted format on internal memory. HMDs are considered to be companions that are worn the whole day and only used when needed. When a user puts on the device in a safe environment, he has to unlock it once through classical password entry. Future versions might be equipped with an eye tracker, which would allow gaze-based password entry [7].

2 The Ubic Framework

The key aim of Ubic is to provide a contribution of practical importance that bridges the gap between digital cryptography and real world applications. We put emphasis on making our solutions as simple as possible and only use well researched and established cryptographic primitives in combination with resource friendly computer vision techniques to allow for easy deployment and seamless integration into existing infrastructures.

The general processing and interaction pipeline of Ubic is shown in Figure 1. Each interaction is initialized by the user scanning the header 2D-barcode (indicated in gray). The header code is composed of the framework header and an application specific header. The former contains the framework version as well as the mode of operation, e.g. identification, content verification, content hiding; the latter is an application specific header, containing information that is relevant for the given application.

Assumptions. The general setting we consider is a user who communicates with a possibly corrupted physical token over an insecure physical channel. In

[2] https://support.google.com/glass/answer/3079857

Table 1. Maximum storage capacity for alphanumeric characters of a version 40 QR code in comparison to the error correction level and the maximum damage it can sustain

EC level	L	M	Q	H
Max. damage (%)	7	15	25	30
Max. characters	4296	3391	2420	1852

this work, we concentrate on the visual channel in connection with HMDs, such as Google Glass. However, our framework can be adapted and extended easily to support other physical channels, such as the auditory channel, if needed. The visual channel is very powerful and key to the vast majority of interactions that humans perform in the real-world. HMDs are personal companions that, in contrast to smartphones, sit right in front of the user's eyes. Google Glass comprises an egocentric camera that allows us to record the visual scene in front of the user, as well as a display mounted in front of the user's right eye. While the developer version that we used could still allow an observer to infer information about the content shown on the display by looking at it from the front, we assume that this is not possible in our attack scenarios. We consider this to be a design flaw of some of the first prototypes, which can be solved easily. Since the display only occludes a small corner of the user's field of view, it could simply be made opaque. We further assume that HMDs are computationally as powerful as smartphones. In practice, this can be achieved by establishing a secure communication channel between the HMD and the user's smartphone.

An *encoder* $\mathsf{E} = (\textsc{Encode}, \textsc{Decode})$ is used to transform digital data from and to a physical representation. We will not mention error-correcting codes explicitly, since we assume them to be a part of the encoder. In particular, our framework uses two-dimensional barcodes, called QR codes [8]. These codes are tailored for machine readability and use Reed-Solomon error correction [9]. Depending on the chosen error correction level, the barcode's capacity differs. Table 1 provides a comparison of their storage capacity for alphanumeric characters and their robustness.

3 Authentication

Our goal was to design an authentication mechanism that allows a user to authenticate himself in front of a token, such as a locked door or an ATM, without revealing his secret credentials to any bystanders who observe the whole authentication process. In addition, even a malicious token should not be able to learn the user's secret credentials. We focused on providing a solution, which is easy to deploy into the current infrastructures, i.e. merely a software update is required, and is as simple and user-friendly as possible.

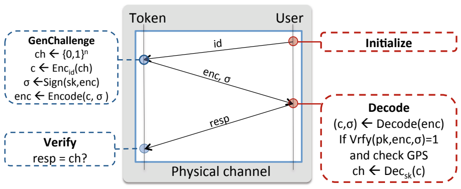

Fig. 2. Visualization of a identification scheme using an optical input device

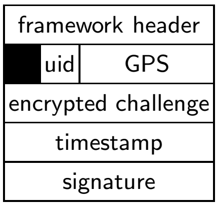

Fig. 3. The identification header composed of the framework and application header

3.1 Threat Model

We consider two different types of adversaries for the authentication scenario. An *active* adversary is able to actively communicate with the user and impersonate the token. He has access to all secrets of the token itself. His aim is to learn a sufficient amount of information about the user's credentials to impersonate him at a later point in time. Note that security against active adversaries implies security against *passive* adversaries, who are only able to observe the data that the user passes to the token during the authentication process. Passive adversaries represent the most common real world adversaries, who can mount attacks like shoulder surfing and skimming. A *man-in-the-middle* adversary is able to misrepresent himself as the token. He is able to communicate with the user and a different token and forward possibly altered messages between the two parties. He does not have the token's secret keys. His aim is to authenticate in front of a different token, while communicating with the user.

Insecurity of current approaches. Clearly, the most common widely deployed solutions, such as those used at ATMs, do not provide sufficient protection against such adversaries. During an authentication process the user's fixed PIN and card information is simply leaked to the adversary, who can then impersonate the user.

3.2 Our Scheme

Let $\Pi_{\mathsf{pke}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA2 secure public-key encryption [10] and $\mathsf{DS} = (\mathsf{Kg}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf})$ a digital signature scheme secure against existential forgery under an adaptive chosen message attack (EU-CMA) [10], where $\mathsf{Kg}_{\mathsf{Sig}}$ is the key generation algorithm, Sig is the signing algorithm, and Vf, the verification algorithm. We assume that the token has knowledge of the user's public key. In the case of an ATM, the key could be given to the bank during registration. Our protocol is a challenge-and-response protocol that we explain with the help of Figure 2. The entire communication between the user and the token uses a visual encoder, which transforms digital information to and from a visual representation. The user initiates the protocol by sending his identifier *id* to the token. The challenger retrieves the corresponding public key from a trusted database, checks the validity of the key, and encrypts a randomly generated challenge $\mathsf{ch} \leftarrow \{0,1\}^{\mathsf{n}}$ using the public-key encryption scheme Π_{pke}. The application header for the identification scenario can be seen in Figure 3. It contains a token identifier (tid), a user identifier (uid), the encrypted challenge, a timestamp, and the token's GPS location. The application header is signed with DS by the token and the signature is appended to the application header. It then generates a QR code consisting of the framework, and the application header.

The resulting QR code is displayed to the user, who decodes the visual representation with his HMD, parses the header information, checks the validity of the signature, the date of the timestamp, whether his location matches the given location, and decrypts the encrypted challenge to obtain ch. The user sends back ch to the token to conclude the authentication process. In the case of an ATM or a locked door, the last step can be done via a key pad. Choosing the length of the challenge is a trade-off between security and usability.

Security Analysis. Due to page constraints, we only provide an informal reasoning, showing that none of our three adversaries can be successful. Note that security against the active adversary already implies security against a passive adversary. Since we assumed that $\Pi_{\mathsf{pke}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is secure against chosen-ciphertext attacks, an adversary is not able to infer any information about the plaintext, i.e. the encrypted PIN, from the given ciphertext, even if he is able to obtain encryptions and decryptions for messages of his choice. This ensures that an (active) adversary can only guess the challenge, since he effectively plays the CCA2 game. To prevent man-in-the-middle attacks, we use an idea called authenticated GPS coordinates, recently introduced by Marforio, Karapanos, and Soriente [11]. We assume that the man-in-the-middle attack is perfomed on two tokens that are at different locations. Recall that each token signs its challenges along with its own GPS location. An adversary is not able to simply forward these challenges between two tokens, since the user, upon receiving a challenge, verifies the signature of the challenge and compares its own location to the signed location. Hence, such an adversary would need to break the unforgeability of DS to be able to forward challenges that will be accepted by the user.

4 Content Verification

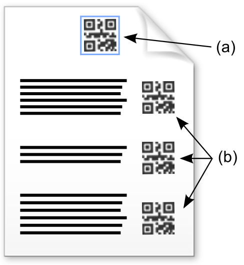

Fig. 4. The *VeriDoc* document format

The goal of our content verification functionality is to enable the generation and verification of physical documents, such as receipts or paychecks, with mathematically proven unforgeability guarantees. In particular, the validity of such documents should be verifiable in a secure, user-friendly, and robust fashion. The combination of physical documents with digital signatures is a challenging task for several reasons. Firstly, the document's content

```
Algorithm 1: Signing
  input  : $m, \mathsf{sid}, \mathsf{sk}, \mathsf{layout}$
  output: Header token $\textsc{Token}_H$ and ;
          side tokens $\textsc{Token}_1, \ldots, \textsc{Token}_\ell$
  $H_f \leftarrow \mathsf{GenFrameworkHeader(Verification)}$
  choose a random $\mathsf{did} \leftarrow \{0,1\}^n$;
  parse $m = 1, \ldots, m_\ell$;
  set $h_H \leftarrow H(H_f, \mathsf{sid}, \mathsf{did}, \ell, \mathsf{layout})$;
  set $\sigma_H \leftarrow \mathsf{Sig}(\mathsf{sk}, h_H)$;
  set $\textsc{Token}_H \leftarrow \textsc{Encode}(H_f, \mathsf{sid}, \mathsf{did}, \ell, \mathsf{layout}, \sigma_H)$;
  Compute QR codes for each message block
  for $i = 1, \ldots, \ell$ do
      $h_i \leftarrow H(\mathsf{did}, m_i, i)$ ;
      $\sigma_i \leftarrow \mathsf{Sig}(\mathsf{sk}, h_i)$ ;
      $\textsc{Token}_i \leftarrow \textsc{Encode}(i, \sigma_i)$;
  return $\textsc{Token}_H, \textsc{Token}_1, \ldots, \textsc{Token}_\ell$
```

Fig. 5. The signing algorithm

```
Algorithm 2: Verify
  input  : Document D
  output: Valid or Invalid
  Verify the document header
  $(H_f, \mathsf{sid}, \mathsf{did}, \ell, \mathsf{layout}) \leftarrow \textsc{Decode}(\textsc{Token}_H)$;
  set $h_H \leftarrow H(H_f, \mathsf{sid}, \mathsf{did}, \ell, \mathsf{layout})$;
  set $\mathsf{vk} \leftarrow \mathsf{PKI(sid)}$;
  return 0 if $\mathsf{Vf}(\mathsf{vk}, h_H, \sigma_H) = 0$;
  Verify each message block
  for $i = 1, \ldots, nb$ do
      $m_i, \textsc{Token}_i \leftarrow \mathsf{OCR}(b_i)$ //see Section 6;
      $h_i \leftarrow H(\mathsf{did}, m_i, i)$ ;
      $i, \sigma_i \leftarrow \textsc{Decode}(\textsc{Token}_i)$;
      return 0 if $\mathsf{Vf}(\mathsf{vk}, h_i, \sigma_i) = 0$ ;
  return 1
```

Fig. 6. The verification algorithm

must be human-readable, which prevents us from using machine-readable visual encodings like QR codes. Secondly, we must be able to transform the human readable content into a digital representation such that we can verify the digital signature. Here, we apply techniques from computer vision such as optical character recognition (OCR). However, OCR has to be performed without any errors and from a practical point of view OCR is very unlikely to succeed without any errors when reading a whole document with an unknown layout. Observe that error-correction techniques cannot be applied, since a contract that says *"Alice gets $100"* is very different from one that says *"Alice gets $1.00"*. Using error-correction one could transform a wrong document into a correct one, which would result in a discrepancy between what the user sees and what is verified. To overcome the aforementioned problems and provide a practical and useable solution, we developed a novel document format, called VeriDoc (see Figure 4). This document facilitates robust document tracking and optical character recognition by encoding additional layout information into it. The layout information is encoded in a header QR code (a) and signatures for each block are encoded into separate QR codes (b).

4.1 Threat Model

Based on the standard EU-CMA notion for digital signature schemes, we consider the following adversary: In the first phase, the query phase, the adversary is able to obtain a polynomial number of (signed) VeriDoc documents for documents of his choice from some user Alice. In the second phase, the challenge phase, the adversary outputs a VeriDoc document D and wins if D verifies under Alice's public key and was not signed by her in the first phase.

4.2 Our Scheme

Let $\mathsf{DS} = (\mathsf{Kg}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf})$ be a signature scheme secure against EU-CMA and H a collision-resistant hash function.

<u>Content Signing:</u> A formal description of the signing algorithm is depicted in Figure 5. It takes the signer's private key sk, his identifier sid, the message $m = m_1, \ldots, m_\ell$ consisting of ℓ blocks as input and the layout information layout. First, the algorithm computes the document header $\textsc{Token}_H$, which comprises of the framework header and the application header. The application header contains the signer's id sid, a randomly generated document id did, the number of message blocks ℓ, and layout. This header is signed and the signature σ_H is appended to the header itself. In the second step, each message block m_i is signed along with did and it's position i. All generated signatures are encoded into QR codes and printed onto the document next to the corresponding message blocks (see Figure 4).

<u>Content Verification:</u> The content verification algorithm, depicted in Figure 6, is given a signed document D consisting of blocks b_i and verifies its validity. The extraction of a message block, the corresponding signature, as well as the underlying computer vision techniques that are used are simplified to $\mathsf{OCR}(\cdot)$ in this description. A description of $\mathsf{OCR}(\cdot)$ will be provided in Section 6. In the first step, the document header $\textsc{Token}_H$ is parsed by the computer vision system. Using the signer's id sid, the corresponding public key vk is obtained from a PKI. Afterwards, the verification algorithm checks the validity of each block. To do so, the algorithm first reads the message block along with its signature $(m_i, \textsc{Token}_i) \leftarrow \mathsf{OCR}(b_i)$, it computes the hash value $h_i \leftarrow H(\mathsf{did}, m_i, i)$, extracts the signature from the corresponding QR code, i.e. $(i, \sigma_i) \leftarrow \textsc{Decode}(\textsc{Token}_i)$ and outputs 0 if the signature is invalid, i.e., if $\mathsf{Vf}(\mathsf{vk}, h_i, \sigma_i) = 1$. If all checks are valid, then the verification algorithm outputs 1.

Security Analysis. We assume that the underlying signature scheme $\mathsf{DS} = (\mathsf{Kg}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf})$ that is used to generate the VeriDoc documents is secure against EU-CMA. This means that an adversary is allowed to obtain signatures on messages of his choice adaptively and he is not able to generate a valid signature for a new message that was not queried to the signing oracle before (except with negligible probability). Furthermore, we assume that the hash function is collision-resistant, meaning that an efficient adversary finds two distinct messages m_0, m_1 that map to the same image $H(m_0) = H(m_1)$ only with negligible probability. In the query phase, the adversary obtains signed tokens for messages of his choice. Note that for each signed token a new random document id $\mathsf{did} \in \{0,1\}^n$ is generated and the header also contains the number of blocks ℓ. This document id prevents so called mix-and-match attacks, where a new valid document is generated by mixing message blocks from other valid documents. Since the id is n-bit long, where n is the security parameter and we consider poly-time adversaries, the probability of two documents having the same id is negligible in n.

Since the signed document header contains the number of message blocks and all blocks are enumerated according to their ordering in the layout, an adversary can neither rearrange, nor remove any message blocks without breaking the unforgeability of the signature scheme. Thus, the resulting VeriDoc document is also existentially unforgeable under chosen message attacks.

4.3 Two-Step Verification

Over the past years a constant increase in digital crime, such as identity theft, has been observed. To counteract these developments, companies like Facebook, Google, Yahoo, and many others allow users to use a technique known as two-factor authentication [12], when using their services. During such an authentication process, an additional layer of security is introduced by requiring a second authentication factor, e.g. a physical token, along with the password. In a similar vein we introduce the *two-step verification technique* that introduces a second step into the process of verifying retrieved content. Consider, for example, a user, who requests his account balance during an online banking session. If the machine that is used is untrusted and possibly even compromised, then the user cannot verify the correctness of the returned balance. To overcome this problem, we use our content verification technique described in Section 4, meaning that in our banking example the account balance is returned together with a visually encoded signature thereof. Using the HMD we parse the signature and the account balance and verify its correctness. An adversary, who wants to convince a user of a false statement, would need to compromise the machine, that is used by him, and the HMD simultaneously, which is considerably harder to achieve in practice. Due to the simplicity of the two-step verification technique, it could easily be integrated into many existing systems immediately.

5 Content Hiding

Motivated by the increasing existence of mobile workplaces, we introduce our content hiding solution. Our goal was to allow users to read confidential documents in the presence of eavesdroppers. HMDs are situated right in front of the user's eye and only he is able to see the displayed content. Confidential documents are printed in an encrypted format and using the HMD an authorized user decrypts the part he is looking at on-the-fly. Applications using this technique are not limited to paper-based documents or tablet computers. Consider an untrusted machine through which a user might want to access some confidential data. Using our content hiding technique, he could obtain the information without leaking it to the untrusted machine. For the sake of clarity and brevity, we describe our technique using public key encryption schemes. In Section 5.3 we show how to realize more complex access structures, such as security clearance hierarchies in office spaces, using predicate encryption schemes.

```
Algorithm 1: Encryption
    input  : $m, ek$
    output : Header token $\textsc{Token}_H$ and ;
             ciphertext tokens $\textsc{Token}_1, \ldots, \textsc{Token}_\ell$
    $H_f \leftarrow \mathsf{GenFrameworkHeader}(\mathsf{Hiding})$;
    choose a random key $k \leftarrow \mathcal{G}(1^\lambda)$;
    compute $\mathsf{key} \leftarrow \mathsf{Enc}(ek, k)$;
    set $\textsc{Token}_H \leftarrow \textsc{Encode}(H_f, \mathsf{key})$;
    Compute encoded ciphertext blocks
    $m_1, \ldots, m_\ell \leftarrow \mathsf{split}(m)$ ;
    for $i = 1, \ldots, \ell$ do
        $c_i \leftarrow \mathcal{E}(k, m_i)$;
        $\textsc{Token}_i \leftarrow \textsc{Encode}(i, c_i)$;
    return $\textsc{Token}_H, \textsc{Token}_1, \ldots, \textsc{Token}_\ell$
```

Fig. 7. The encryption algorithm

```
Algorithm 2: Decryption
    input  : $\textsc{Token}_H, \textsc{Token}_1, \ldots, \textsc{Token}_\ell, dk$
    output : message $m$
    Decode the header
    $(H_f, \mathsf{key}) \leftarrow \textsc{Decode}(\textsc{Token}_H)$;
    compute $k \leftarrow \mathsf{Dec}(dk, \mathsf{key})$;
    Decrypt the ciphertext
    for $i = 1, \ldots, \ell$ do
        $(i, c_i) \leftarrow \textsc{Decode}(\textsc{Token}_i)$;
        $m_i \leftarrow \mathcal{D}(k, m_i)$;
    return $m = m_1, \ldots, m_\ell$
```

Fig. 8. The decryption algorithm

5.1 Threat Model

In this scenario we basically consider the adversary from the standard CCA2 security notion. The adversary is allowed to obtain a polynomial amount of encryptions and decryptions for messages and ciphertexts of his choice from some honest user Alice. At some point the adversary outputs two messages, Alice picks one at random, and encrypts it. The adversary wins if he can guess which message was encrypted with a probability of at least $\frac{1}{2} + \epsilon(n)$, where ϵ is a non-negligible function and n is the security parameter.

5.2 Our Scheme

Let $\Pi_{\mathsf{pke}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA2 secure public key, and $\Pi_{\mathsf{priv}} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ a CCA2 secure private key encryption scheme. To obtain public key encryption scheme with short ciphertexts, we use a hybrid encryption scheme [13]. The basic idea of such a scheme is to encrypt a randomly generated key $k \leftarrow \mathcal{G}(n)$ with the public key encryption scheme and store it in the header. The actual plaintext is encrypted using Π_{priv} with k.

ENCRYPTION: The encryption algorithm is depicted in Figure 7 and works as follows: At first, a randomly chosen *document key* k is encrypted with a public-key encryption scheme under the public key ek of the recipient. A header QR code $\textsc{Token}_H$ is created, which contains the framework header, the encrypted document key. The actual body of the document m is split into message chunks $m_1, \ldots, m_\ell$ and each chunk is encrypted separately using the document key and is then encoded, along with the block id, into a QR code $\textsc{Token}_i$.

DECRYPTION: The decryption algorithm is depicted in Figure 8. Upon receiving a document, the receiver decodes the header QR code, obtains the encrypted document key key. Using his secret key dk, the algorithm recovers the document key $k \leftarrow \mathcal{D}(dk, \mathsf{key})$ and it uses the key to decrypt the document body.

The advantages of representing the document as a sequence of encrypted blocks is twofold. Firstly, it allows the user to only decrypt the part of the encrypted document body that he is currently looking at without the need to scan the whole document first. Furthermore, the encrypted documents are robust to damage, meaning that even if a part of it is broken or unreadable, we are still able to decrypt the remaining undamaged ciphertext blocks as long as the document header is readable. Choosing the size of the message blocks is a trade-off between space and robustness. The bigger the message blocks are, the more plaintext is lost once a single QR code is not readable anymore. The smaller they are, the more QR codes are required, hence the more space is needed to display them.

Security Analysis. It is well known that using the hybrid argument proof technique [10] the CCA2 game, where the adversary outputs two distinct messages in the challenge phase, is equivalent to a CCA2 game where the adversary outputs two message vectors of polynomial length. The security of our scheme directly follows from this observation.

5.3 Extending Content Hiding to Support Fine-Grained Access Control

Using public-key encryption in our content hiding scheme allows us to encrypt documents for certain recipients. In companies or organizations, however, it is more desirable to encrypt documents, such that only employees with certain security clearances can read certain enrypted documents. Ubic allows to encrypt documents, such that only users with certain security clearances can read them. Therefore, we replace the public-key encryption scheme by a *predicate* encryption scheme [14]. Loosely speaking, in a predicate encryption scheme, one can encrypt a message M under a certain attribute $I \in \Sigma$ using a master public key mpk where Σ is the universe of all possible attributes. The encryption algorithm outputs a ciphertext that can be decrypted with a secret key sk_f associated with a predicate $f \in \mathcal{F}$, if and only if I fulfills f, i.e., $f(I) = 1$, where $\mathcal{F}$ is the universe of all predicates.

Next, we explain the security notion of predicate encryption, called *attribute-hiding*, with the following toy example. Consider the scenario where professors, students, and employees are working at a university and by $Prof$, Emp, and $Stud$ we denote the corresponding attributes. Every member of a group will be equipped with a secret key sk_f such that f is either the predicate $\mathsf{mayAccProf}$, $\mathsf{mayAccEmp}$, or $\mathsf{mayAccStud}$. We use the toy policy that professors may read everything and employees and students may only read encryptions created using Emp and $Stud$, respectively. Now, attribute-hiding states that a file *file* which is encrypted using the attribute $Prof$, can not be decrypted by a student equipped with $sk_{\mathsf{mayAccStud}}$ and the student also can not tell with which attribute *file* was encrypted (except for the fact that it was not $Stud$). Furthermore, even a professor does not learn under which attribute *file* was encrypted, she only learns the content of the file and nothing more.

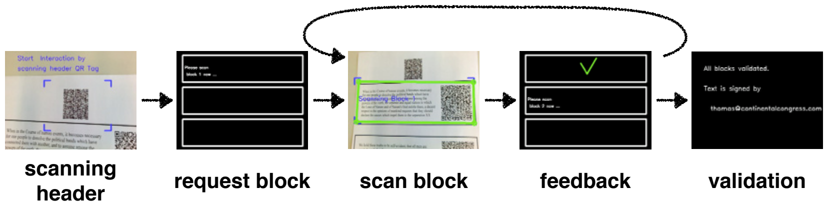
Start interaction by scanning header QR Tag
Please scan block 1 now ...
Scanning Block 1
Please scan block 2 now ...
All blocks validated.
Text is signed by
thomas@continentalcongress.com
scanning header
request block
scan block
feedback
validation

Fig. 9. Interaction cycle with VeriDoc. The user initiates the interaction by scanning the header QR code at the top of the document. After sequential scanning of each content block, the user is informed if the document was verified or not. The black screens are what the user sees on the Google Glass display. They cover the whole screen but only a small part of the users view.

Extending Our Scheme. We extend our scheme to also support fine grained access control by replacing the public-key encryption scheme with a predicate encryption scheme. Thus, the user encrypting the message in addition chooses an attribute $I \in \Sigma$ that specifies which users can decrypt the message. Formally, our encryption algorithm is almost the same as described in Figure 7, but the public-key encryption step is replaced with $c \leftarrow \mathsf{PrEnc}(mpk, I, k)$, where mpk is a master public key that works for all attributes. The only difference in the decryption algorithm is that instead of using the public-key decryption algorithm Dec, we are now running the decryption algorithm of the predicate encryption scheme $\mathsf{PrDec}(sk_f, c)$ and the user can only decrypt if $f(I) = 1$.

Efficient Implementation. Our implementation is based on the predicate encryption scheme due to Katz, Sahai, and Waters [14] (see Section A for a formal description of the scheme). However, for efficiency reasons, we did not implement the scheme in composite order groups, but adapted the transformation to prime order groups as suggested by Freeman [15].

6 The VeriDoc Interface

In the following, we describe our document format VeriDoc. A high-level overview of the document scanning process is shown in Figure 9. Throughout this process we provide visual feedback to make the scanning process transparent to the user. As already described, the user initiates the document verification by scanning the header code of the document. Amongs other information, the header code contains the layout information. This information contains additional information about the document that facilitates the scanning process. In particular, this information contains the used font, the aspect ratio of each message block, and the document language. After scanning the header code, the user is asked to scan the message blocks. We display brackets on the HMD to help the user to position the camera properly over the text block Accurate alignment and content extraction is further facilitated by a computer vision subsystem as described below. After each scanned block, its content is extracted and verified against the

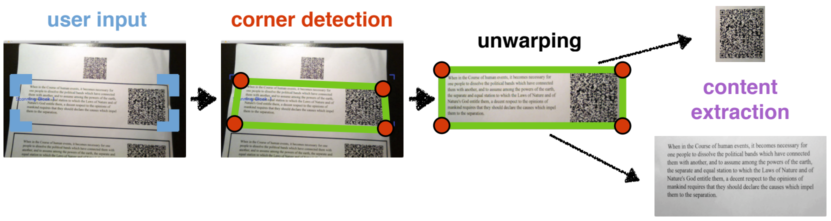

Fig. 10. Vision subsystem to assist the user in working with VeriDoc: The user points the front-facing camera roughly at the VeriDoc document, the system detects the four corners of the first content block and snaps the locations of the brackets to them, and the system unwarps and extracts the content of that block.

signature encoded in the QR code. The user is informed about the validity of each text block and once all blocks of a given document are scanned the system informs the user if the document, as a whole, was successfully verified.

Alignment and Content Extraction: To assist the user in scanning VeriDoc document content we provide a refinement procedure that allows the user to roughly indicate relevant text blocks, but still provide the required accuracy for the computer vision processing pipeline (see Figure 10). On the very left, a typical user interaction is depicted showing a coarse alignment of the brackets with the first text block. We proceed by a corner detection algorithm and snap the locations of the brackets provided by the user to the closest corners. We use the Harris corner measure M to robustly detect corners [16]:

$$A = g(\sigma_I) * \begin{pmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{pmatrix} \tag{1}$$

$$M = \det(A) - \kappa \,\mathrm{trace}(A)^2 \tag{2}$$

where I_x and I_y are the spatial image derivatives in x and y direction, σ_D smoothing of the image with the detection scale and σ_I smoothing the response with the integration scale and $\kappa = 0.04$ according to best practice. Intuitively, the pre-smoothing with σ_D eliminates noise and allows detection of corners at a desired scale [17] while the smoothing σ_I suppresses local maxima in the response function. In order to be robust to the choice of these scales, we employ the multi-scale harris detector that finds corners across multiple scales [18].

The second image at the top shows a visualization of the closest corner and the box spanned by them in green. Under the assumption of a pinhole camera model as well as a planar target (documents in our case), we can compute a homography $H \in \mathbb{R}^{3\times3}$ in order to undo the perspective transformation under which the content is viewed. The matrix H relates the points under the perspective project p' to the points under an orthogonal viewing angle p by

$$p' = Hp \tag{3}$$

where $p, p' \in \mathbb{R}^3$ are given in homogeneous coordinates. As our interface has determined the 4 corners that each specify a pair of p and p', we have sufficient information to estimate matrix H.

The third image from the left in Figure 10 shows the content after unwarping and cropping. Using the information on the ratio between text and code contained in the header, we now split the content area into text and the associated QR code. In a last step, the QR code is decoded, the signature extracted, and the text area is further processed using OCR in combination with the font and language information from the header code.

7 Prototype Implementation

We provide a prototype implementation, written in Java, of our Ubic framework on the Google Glass device. The device runs Android 4.0.4 as its underlying operating system, features a 640×360 optical head-mounted display as well as an egocentric camera with a resolution of 1280×720 pixels. Our current developer version only features an embedded microcontroller with 1.2 GHz and 1GB of memory.

We used the Bouncy Castle Crypto API 1.50 [19] and the Java Pairing-Based Cryptography Library 2.0.0 (JPBC) [20] to implement all required cryptographic primitives. In particular, we used SHA-1 as our collision-resistant hash function, SHA1+RSA-PSS as our signature, AES-256 in CTR-Mode as our private-key encryption, and RSA-OAEP with 2048 bit long keys as our public-key encryption scheme. For our predicate encryption scheme, we use a MNT curve [21] with a security parameter of 112 according to the NIST recommendations [22]. For the computer vision part of our framework, we used the OpenCV 2.4.8 image processing library [23], and QR codes are being processed with the barcode image processing library zxing 1.7.6 [24]. For optical character recognition, we used the Tesseract OCR engine [25].

8 Related Work

Head-mounted displays, such as Google Glass, have raised strong privacy concerns in the past and recent publications [26,27,28] have tried to address these issues. In [26], the authors suggest that operating systems should provide high-level abstractions for accessing perceptual data. Following this line of work, [27] proposes a system, which makes a first step towards providing privacy guarantees for sensor feeds. There, applications access the camera through a new interface rather than accessing the camera directly. Depending on the application's permissions, the camera is pre-processed with different sets of image filters, which aim to filter sensitive information. In [28], the notion of recognizers is introduced. Rather than passing a filtered sensor feed to the requesting application, they provide a set of recognizers that fulfil the most common tasks, such as face detection or recognition. Applications obtain permissions for certain recognizer and can request the output of certain computations on the sensor feed. In contrast to our work, this line of research regards the device as a threat.

Another line of research concentrates on establishing trust between devices based on the visual channel [29,30,31]. In [31], for instance, the visual channel is used for demonstrative authentication, where two devices authenticate themselves towards each other by basing their trust on the visual channel between them. One possible application for this authentication mechanism is access points with QR codes printed onto them. The user scans the QR code to authenticate the access point.

In [32], a survey of different techniques for scanning and analyzing documents with the help of cameras, cell phones, and wearable computers is provided. The survey shows that even though constant progress is made, current methods are not robust enough for real world deployment. Ubic tackles this problem in a different way by facilitating the task of scanning documents by encoding additional information into them.

9 Conclusion

We presented Ubic, a framework that makes an important step towards a tighter integration of digital cryptography in real-world applications. Using HMDs in combination with established cryptographic primitives and resource friendly computer vision techniques, we provide users with more security and privacy guarantees in a wide range of common real-world applications. We present user-friendly, easy-to-use solutions for authentication, content verification, content hiding, that can seamlessly be integrated into the current infrastructure. We hope that our work will stimulate further research investigating the possibilities of combining ubiquitous computing technologies with cryptographic primitives in a user-friendly fashion.

Acknowledgements. Andreas Bulling and Mario Fritz are supported by a Google Glass Research Award. Work of Dominique Schröder and Mark Simkin was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy, and Accountability (CISPA; see `www.cispa-security.org`). Dominique Schröder is also supported by an Intel Early Career Faculty Honor Program Award.

References

1. News, B.: Cash machines raided with infected usb sticks (2013)
2. Bankrate: Skimming the cash out of your account (2002)
3. Times, N.Y.: Target missed signs of a data breach (2014)
4. Telegraph, T.: Mind how you move that chair - it's hot hot-desking is a growing trend, bringing a new culture writes violet johnstone (2002)
5. House, T.W.: Bring your own device (2012)
6. for Visual Data Security, E.A.: Visual Security White Paper (2012)
7. Kumar, M., Garfinkel, T., Boneh, D., Winograd, T.: Reducing shoulder-surfing by using gaze-based password entry. In: Proceedings of the 3rd Symposium on Usable Privacy and Security, SOUPS 2007, pp. 13–19. ACM (2007)

8. International Organization for Standardization: Information technology — automatic identification and data capture techniques — qr code 2005 bar code symbology specification (2006)
9. Wicker, S.B.: Reed-Solomon Codes and Their Applications. IEEE Press, Piscataway (1994)
10. Katz, J., Lindell, Y.: Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series). Chapman & Hall/CRC (2007)
11. Marforio, C., Karapanos, N., Soriente, C., Kostiainen, K., Capkun, S.: Smartphones as practical and secure location verification tokens for payments. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2014 (2014)
12. Van Rijswijk, R.M., Van Dijk, J.: Tiqr: A novel take on two-factor authentication. In: Proceedings of the 25th International Conference on Large Installation System Administration, LISA 2011, p. 7. USENIX Association (2011)
13. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. 33(1), 167–226 (2004)
14. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
15. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010)
16. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of the 4th Alvey Vision Conference, pp. 147–151 (1988)
17. Lindeberg, T.: Scale-Space Theory in Computer Vision. Kluwer Academic Publishers, Norwell (1994)
18. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(10), 1615–1630 (2005)
19. The Legion of the Bouncy Castle: Lightweight Cryptography API (Release 1.50)
20. De Caro, A., Iovino, V.: jpbc: Java pairing based cryptography. In: Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011, Kerkyra, Corfu, Greece, June 28-July 1, pp. 850–855 (2011)
21. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for fr-reduction (2001)
22. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for Key Management Part 1: General (Revision 3). Technical report (July 2012)
23. Bradski, G.: Open source computer vision library (opencv) (2000)
24. ZXing: ZXing Multi-format 1D/2D barcode image processing library (2012)
25. Smith, R.: An overview of the tesseract ocr engine. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, ICDAR 2007, vol. 2, pp. 629–633. IEEE Computer Society, Washington, DC (2007)
26. D'Antoni, L., Dunn, A., Jana, S., Kohno, T., Livshits, B., Molnar, D., Moshchuk, A., Ofek, E., Roesner, F., Saponas, S., Veanes, M., Wang, H.J.: Operating system support for augmented reality applications. In: Proceedings of the 14th USENIX Conference on Hot Topics in Operating Systems, HotOS 2013, p. 21. USENIX Association, Berkeley (2013)
27. Jana, S., Narayanan, A., Shmatikov, V.: A scanner darkly: Protecting user privacy from perceptual applications. In: IEEE Symposium on Security and Privacy, pp. 349–363. IEEE Computer Society (2013)

28. Jana, S., Molnar, D., Moshchuk, A., Dunn, A., Livshits, B., Wang, H.J., Ofek, E.: Enabling Fine-Grained Permissions for Augmented Reality Applications With Recognizers. In: 22nd USENIX Security Symposium (USENIX Security 2013), Washington DC (August 2013)
29. Starnberger, G., Froihofer, L., Goeschka, K.M.: Qr-tan: Secure mobile transaction authentication. In: 2012 Seventh International Conference on Availability, Reliability and Security, pp. 578–583 (2009)
30. Saxena, N., Ekberg, J.E., Kostiainen, K., Asokan, N.: Secure device pairing based on a visual channel. In: 2006 IEEE Symposium on Security and Privacy, pp. 306–313 (2006)
31. Mccune, J.M., Perrig, A., Reiter, M.K.: Seeing-is-believing: Using camera phones for human-verifiable authentication. In: IEEE Symposium on Security and Privacy, pp. 110–124 (2005)
32. Liang, J., Doermann, D., Li, H.: Camera-based analysis of text and documents: a survey. International Journal on Document Analysis and Recognition 7, 84–104–104 (2005)

A Predicate Encryption

For completeness, we recall the predicate encryption scheme due to Katz, Sahai, and Waters [14].

Definition 1 (Predicate Encryption). *A* predicate encryption scheme *for the universe of predicates and attributes $\mathcal{F}$ and Σ, respectively, is a tuple of efficient algorithms $\Pi_{\mathsf{PE}} = (\mathsf{PrGen}, \mathsf{PrKGen}, \mathsf{PrEnc}, \mathsf{PrDec})$, where the generation algorithm* PrGen *takes as input a security parameter 1^λ and returns a master public and a master secret key pair (mpk, psk); the key generation algorithm* PrKGen *takes as input the master secret key psk and a predicate description $f \in \mathcal{F}$ and returns a secret key sk_f associated with f; the encryption algorithm* PrEnc *takes as input the master public key mpk, an attribute $I \in \Sigma$, and a message m and it returns a ciphertext c; and the decryption algorithm* PrDec *takes as input a secret key sk_f associated with a predicate f and a ciphertext c and outputs either a message m or $\perp$.*

A predicate encryption scheme Π_{PE} is *correct* if and only if, for all λ, all key pairs $(mpk, psk) \leftarrow \mathsf{PrGen}(1^\lambda)$, all predicates $f \in \mathcal{F}$, all secret keys $sk_f \leftarrow \mathsf{PrKGen}(psk, f)$, and all attributes $I \in \Sigma$ we have that (i) if $f(I) = 1$ then $\mathsf{PrDec}(sk_f, \mathsf{PrEnc}(mpk, I, m)) = m$ and (ii) if $f(I) = 0$ then $\mathsf{PrDec}(sk_f, \mathsf{PrEnc}(mpk, I, m)) = \perp$ except with negligible probability.

The KSW Predicate Encryption Scheme. The scheme is based on composite order groups with a bilinear map. More precisely, let $N = pqr$ be a composite number where p, q, and r are large prime numbers. Let $\mathbb{G}$ be an order-N cyclic group and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map. Recall that e is *bilinear*, i.e., $e(g^a, g^b) = e(g, g)^{ab}$, and *non-degenerate*, i.e., if $\langle g \rangle = \mathbb{G}$ then $e(g, g) \neq 1$. Then, by the chinese remainder theorem, $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$ where $\mathbb{G}_s$ with $s \in \{p, q, r\}$ are the s-order subgroups of $\mathbb{G}$. Moreover, given a generator g for $\mathbb{G}$, $\langle g^{pq} \rangle = \mathbb{G}_r$,

$\langle g^{pr}\rangle = \mathbb{G}_q$, and $\langle g^{qr}\rangle = \mathbb{G}_p$. Another insight is the following, given for instance $a \in \mathbb{G}_p$ and $b \in \mathbb{G}_q$, we have $e(a, b) = e((g^{qr})^c, (g^{pr})^d) = e(g^{rc}, g^d)^{pqr} = 1$, i.e., a pairing of elements from different subgroups cancels out. Finally, let $\mathcal{G}$ be an algorithm that takes as input a security parameter 1^λ and outputs a description $(p, q, r, \mathbb{G}, \mathbb{G}_T, e)$. We describe the algorithms PrGen, PrKGen, PrEnc, and PrDec in the sequel.

Algorithm $\mathsf{PoGen}(1^\lambda, n)$ *and* $\mathsf{PrGen}(1^\lambda, n)$. First, the algorithm runs $\mathcal{G}(1^\lambda)$ to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, e)$ with $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Then, it computes g_p, g_q, and g_r as generators of $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, respectively. The algorithm selects $R_0 \in \mathbb{G}_r$, $R_{1.i}, R_{2,i} \in \mathbb{G}_r$ and $h_{1,i}, h_{2,i} \in \mathbb{G}_p$ uniformly at random for $1 \leq i \leq n$. $(N = pqr, \mathbb{G}, \mathbb{G}_T, e)$ constitutes the public parameters. The public key for the predicate-only encryption scheme is

$$opk = (g_p, g_r, Q = g_q \cdot R_0, \{H_{1,i} = h_{1,i} \cdot R_{1,i}, H_{2,i} = h_{2,i} \cdot R_{2,i}\}_{i=1}^n)$$

and the master secret key is $osk = (p, q, r, g_q, \{h_{1,i}, h_{2,i}\}_{i=1}^n)$. For the predicate encryption with messages, the algorithm additionally chooses $\gamma \in \mathbb{Z}_N$ and $h \in \mathbb{G}_p$ at random. The public key is

$$mpk = (g_p, g_r, Q = g_q \cdot R_0, P = e(g_p, h)^\gamma, \{H_{1,i} = h_{1,i} \cdot R_{1,i}, H_{2,i} = h_{2,i} \cdot R_{2,i}\}_{i=1}^n)$$

and the master secret key is $psk = (p, q, r, g_q, h^{-\gamma}, \{h_{1,i}, h_{2,i}\}_{i=1}^n)$.

Algorithm $\mathsf{PoKGen}(osk, \vec{v})$ *and* $\mathsf{PrKGen}(psk, \vec{v})$. Parse $\vec{v}$ as $(v_1, \ldots, v_n)$ where $v_i \in \mathbb{Z}_N$. The algorithm picks random $r_{1,i}, r_{2,i} \in \mathbb{Z}_p$ for $1 \leq i \leq n$, random $R_5 \in \mathbb{G}_r$, random $f_1, f_2 \in \mathbb{Z}_q$, and random $Q_6 \in \mathbb{G}_q$. For the predicate-only encryption scheme, it outputs a secret key

$$osk_{\vec{v}} = \begin{pmatrix} K_0 = R_5 \cdot Q_6 \cdot \prod_{i=1}^n h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}, \\ \{K_{1,i} = g_p^{r_{1,i}} \cdot g_q^{f_1 \cdot v_i}, K_{2,i} = g_p^{r_{2,i}} \cdot g_q^{f_2 \cdot v_i}\}_{i=1}^n \end{pmatrix}.$$

For the predicate encryption scheme with messages, the secret key $sk_{\vec{v}}$ is the same as $osk_{\vec{v}}$ except for

$$K_0 = R_5 \cdot Q_6 \cdot h^{-\gamma} \cdot \prod_{i=1}^n h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}.$$

Algorithm $\mathsf{PoEnc}(opk, \vec{x})$ *and* $\mathsf{PrEnc}(mpk, \vec{x}, m)$. Parse $\vec{x}$ as $(x_1, \ldots, x_n)$ where $x_i \in \mathbb{Z}_N$. The algorithm picks random $s, \alpha, \beta \in \mathbb{Z}_N$ and random $R_{3,i}, R_{4,i} \in \mathbb{G}_r$ for $1 \leq i \leq n$. For the predicate-only encryption scheme, it outputs the ciphertext

$$C = \begin{pmatrix} C_0 = g_p^s, \{C_{1,i} = H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}, \\ C_{2,i} = H_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot R_{4,i}\}_{i=0}^n \end{pmatrix}.$$

For the predicate encryption scheme with messages notice that $m \in \mathbb{G}_T$. The ciphertext is

$$C = \begin{pmatrix} C' = m \cdot P^s, C_0 = g_p^s, \\ \{C_{1,i} = H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}, \\ C_{2,i} = H_{2,i}^s \cdot Q^{\beta \cdot x_i} \cdot R_{4,i}\}_{i=0}^n \end{pmatrix}.$$

Algorithm $\mathsf{PoDec}(osk_{\vec{v}}, C)$ *and* $\mathsf{PrDec}(sk_{\vec{v}}, C)$*.* The predicate-only encryption outputs whether the following equation is equal to 1

$$e(C_0, K_0) \cdot \prod_{i=1}^{n} e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}).$$

The predicate encryption scheme with messages outputs the result of the following equation

$$C' \cdot e(C_0, K_0) \cdot \prod_{i=1}^{n} e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}).$$