Ubic: Bridging the gap between digital cryptography and the physical world

Mark Simkin
Saarland University
Germany

Andreas Bulling
Max Planck Institute
for Informatics
Germany

Mario Fritz
Max Planck Institute
for Informatics
Germany

Dominique Schröder
Saarland University
Germany

Abstract—Advances in computing technologies increasingly blur the boundary between the digital domain and the physical world. Even though the research community has developed a large number of cryptographic primitives and has shown their usability in all-digital communication, many of them have not yet made their way into the real world. This work aims to make another step towards a tighter integration of digital cryptography into real world interactions. We first propose a security model that aims to model real-world attackers that have access to the physical channels between an actor and a token. We then describe Ubic, a framework that allows users to bridge the gap between digital cryptography and the physical world. The framework covers key cryptographic primitives, such as secure identification, document verification using a novel secure physical document format based on QR codes, as well as content hiding. Finally, we present an open-source implementation of our framework that is available to the community for further study and future extension.

I. INTRODUCTION

Wearable computing has seen the recent advent of egocentric vision devices and head-mounted displays (HMD), such as Google Glass[1] or Oculus Rift[2], motion sensors, such as the FitBit system[3] or the Kinect[4] and wireless technologies, such as RFID and Bluetooth in portable handheld devices. Similarly, augmented reality, where virtual objects are superimposed on top of a user’s real world view or where users can interact with a virtual world, has attracted increasing interest. All of these technologies increasingly blur the boundaries between the physical and the digital world.

In contrast, most cryptographic primitives, like signatures, encryption or authentication protocols, are still primarily used in the digital domain. Although the development of many of these primitives was initially motivated by real-world problems, we barely make use of them in a broad range of practical real-world applications, such as when signing contracts or withdrawing money from an ATM. Instead, we are using primitives that work well in the analogue world but that do not provide the same level of security or trust as their digital counterparts.

Consider the example of signatures. For those we have provably secure realizations but signatures in the physical world are typically hand-written and therefore have several drawbacks. Most importantly they are not unforgeable and an average user has no reliable way of verifying a hand-written signature. Although a professional forensic document examiner can provide a careful analysis of such a signature he might still fail to detect a good forgery. A second problem with hand-written signatures is the legal requirement to store business records. German companies, for example, generate about 35 billion business documents each year and they all have to be stored for at least ten years. Companies try to address the problem by scanning the documents and only storing their digital version. However, this practice is a reoccurring object of legal dispute [1]. Digital signatures on the other hand are formally defined, have a proof of security and provide means for verifying existing signatures. Being able to use them in connection with physical documents in a convenient way would solve the problems mentioned above.

Identification is another common important real-world ap-
plication that has a well researched digital counterpart. Con- sider for example our current method of withdrawing money from an ATM, where a user uses his bank card and a fixed PIN to identify himself. For an attacker it is sufficient to observe one PIN entry and then obtain the users bank card to get full access to the bank account. In contrast to this, challenge- response protocols allow a user to authenticate himself in such a way that an attacker cannot infer the users secret from observing the protocol execution.

The aforementioned problems underline the need to bridge the gap between the real world and digital cryptography. At the same time they highlight the considerable potential of bringing digital cryptology closer to the real world. These technologies will allow users to enjoy the advantages of digital cryptography and fulfill their security and privacy needs in all their interactions in the real world.

A. Contributions

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 applications (see Figure 1). Ubic addresses key limitations of current systems by relying on an egocentric computer vision system in combination with a head-mounted display. We introduce the physical attacker model (PAM) to model more realistic real-world attackers that have access to the physical channels between the actor and the token, such as an ATM or a signed contract. Specifically, we model the actor and the token as abstract entities that communicate over an insecure physical channel, such as the visual or auditory channel. In our model the attacker is able to eavesdrop, intercept, and alter communication on the physical channel between the actor and the token and he is able to either corrupt the token or the actor. Ubic offers the following key functionalities:

**Identification**: Our identification protocol combines an HMD and a challenge-response protocol in order to allow the actor to identify himself to a token, such as an ATM or a locked door. This protocol does not reveal the login credentials to any bystander who observes the communication between them. Here, the actor reveals his identity to the token. The token generates a random challenge that has to be solved by the actor with the help of his secret key that is stored in his HMD.

**Content Verification**: We propose a new document format, VeriDocChat that allows for robust document tracking and optical character recognition, and contains a digital signature of its content. Using the HMD, an actor can conveniently and reliably verify the validity of the document’s content. In addition, the use of digital signatures would allow companies to only store a digital version of their documents.

**Two-Factor Verification**: Based on the signature functionality described above, we introduce two-factor verification of content. Imagine, during a online banking session, a user might request his current account balance. This balance is then returned 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**: Taking the physical channel between the user and the token into account enables us to consider new scenarios, where a user might want to read confidential documents in public spaces securely with the help of HMDs. The text printed on the physical document is encrypted so that a nosy bystander cannot learn anything about the document’s content. The HMD is then used to decrypt those parts of the text that the user is currently looking at. In addition, using a predicate encryption scheme, we allow content hiding with fine-grained access control. That is, rather than encrypting a document for a certain recipient, we encrypt it for a set of users that fulfill certain attributes. One possibly interesting application of such a fine grained content hiding approach are paper-based offices, where different employees have different clearances and one would like to encrypt documents in such a way that only certain employees can read their contents.

II. The Ubic Framework

The key aim of Ubic is to bridge 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 2. Every interaction is initialized by the actor scanning a header QR tag (indicated in gray). The header tag is composed of the framework header and an application header. The former contains information about the frameworks version as well as the mode of operation, e.g. identification, content verification, content hiding that will be used; the latter is an application specific header, containing information that is relevant for the given application. In the identification scenario, after scanning the header tag, the user obtains an encrypted PIN and has to enter the PIN that is decrypted and shown to him by the HMD. For content verification and content hiding the user is guided through a sequence of scanning operations to validate or decrypt the document. After each operation the user receives visual feedback on whether the scanning operation was successful, i.e. whether the content could be verified or decrypted. If this was not the case or if there was any error, the user can choose to scan the respective block or tag again.

A. Assumptions

The general setting we consider is a user who communicates with a possibly corrupted physical token over an insecure physical channel. In this work we concentrate on visual
Figure 2. Overview of the Ubic processing and interaction pipeline for the different operation modes: identification (a), content verification (b), and content hiding (c). For all operation modes the user is first instructed to scan the header QR tag (indicated in gray). For identification he then has to enter the PIN shown to him; for content verification and hiding the user is guided through a sequence of scanning operations to validate or decrypt the document.

Table I

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

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 with computing devices 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 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. 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. Using the smartphone directly without the HMD is not sufficient, since there is still a physical channel between the display and the human eye that can be observed by an attacker. Such an approach would therefore be vulnerable to eavesdropping techniques like shoulder surfing.

An encoder \( E = (\text{ENCODE}, \text{DECODE}) \) 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 bar codes, called QR-Codes \(^2\). These codes are tailored for machine readability and use Reed-Solomon error correction \(^3\) with four error correction levels. Table I provides a comparison of their storage capacity for alphanumeric characters and their robustness.

B. Key Storage

All secret keys and certificates are stored in an encrypted key storage on the device. Using a secret master key the user is able to unlock this key storage and use the keys in it. A user might, for instance, unlock the device when putting it on and logout when he takes it off. This way an attacker, who at some point might gain access to the HMD will not obtain the user’s secret keys. One issue here is that the key storage is unlocked for most of the time. To circumvent this, the user would need to be able to unlock this storage when needed, but currently there is no straightforward way of achieving this with Google Glass. Future versions of HMDs might be stocked with an eye tracker. This would allow the user to unlock the key storage when needed by using a gaze-based password entry \(^4\).

III. IDENTIFICATION

The goal of our identification scheme is to allow an actor 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 authentication process, or even the token itself. We assume that the token has knowledge of the actor’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 3. The entire communication between the actor and the token uses a visual encoder, which transforms digital information to and from a visual representation. The actor 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 \( ch \leftarrow \{0,1\}^n \) using a public-key encryption scheme \( \Pi_{pke} = (Gen, Enc, Dec) \) and signs the entire header with a digital signature scheme \( DS = (Kg_{Sig}, Sig, Vf) \), where \( Kg_{Sig} \) is the key generation algorithm, \( Sig \) is the signing algorithm, and \( Vf \), the verification algorithm. It then generates a QR-Code consisting of the framework, and the application header for the authentication scenario. Based on the application information in the framework header, the application header is parsed. A detailed description of the application header for the identification scenario can be seen in Figure 4. It contains the applications version, the actor’s id (aid), the token’s id (tid), the encrypted challenge and a timestamp. The application header is signed by the token and the signature is appended to the application header. The resulting QR-Code is then displayed to the actor, who decodes the visual representation, parses the header information, checks the validity of the signature, the date of the timestamp and decrypts the encrypted challenge to obtain \( ch \). The actor 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.

### A. Threat Model

We consider a polynomially bounded adversary in the PAM. That is, he is in control of the physical channel between the actor and the token and he is able to misrepresent himself as one of the parties and at the end of the game he tries to authenticate himself as a valid actor in front of an honest token. This intuition is captured in the following two-stage game consisting of a learning and a challenge phase:

- **a) Learning Phase:** In the learning phase, the adversary impersonates a token, who communicates to an honest actor. The attacker can choose the challenges that are sent to the actor adaptively and in addition, he controls the physical channel, which means that the complete transcript of the communication is known to him.

- **b) Challenge Phase:** In the challenge phase the adversary now plays the role of the actor playing against an honest token. In this phase, the adversary runs the identification protocol with an honest challenger. He wins if he successfully authenticates himself in front of the challenger. In case the attackers identification attempt fails, he may either go back to the learning phase or run another identification protocol with the challenger.

It is important to note that most common and widely deployed identification schemes, such those used by ATMs or locked doors, are insecure with respect to our security definition. In the first phase the attacker would simply obtain all information from the actor’s card and execute the authentication protocol once to obtain the secret PIN. In the second phase the attacker could simply use the information from the first phase to forge the card and use the PIN to successfully authenticate himself.

---

**Figure 4.** The identification header and the corresponding QR Code. The QR-Code on the right is an exemplary identification header. Apart from the framework header it contains the public-key algorithm (alg) that was used, the actor’s id (aid), the encrypted challenge, the timestamp, and the signature of the header itself.

### B. Security Analysis

We discuss the security of our system based on general cryptographic assumptions without relying on any concrete instantiations. We assume that the public-key encryption scheme \( \Pi_{pke} = (Gen, Enc, Dec) \) is secure against chosen-ciphertext attacks (CCA-Secure). Roughly speaking, this means that an adversary is not able to infer any information about a plain text from a given ciphertext, even if he is able to obtain encryptions and decryptions for messages of his choice.

Assuming that all components fulfill the described assumptions we now argue why our identification system is secure. First we assume that the keys have been generated honestly. In the first stage, the learning phase, the adversary controls the token and all communication channels. Therefore he is able to generate challenges of his choice and send their encryptions to the actor. In addition he can simply send maliciously generated ciphertexts.

Now, at some point the adversary enters the challenge phase, i.e. he starts misrepresenting himself in front of an honest token. The adversary initializes the protocol by sending an id to the token. Since the token verifies the validity of the the public-key that corresponds to id, the check guarantees that the adversary is attacking a real account and does not try to withdraw money from a fake account. Upon receiving an encrypted challenge the adversary can only guess with
a success probability of $2^{-n}$, where $n$ is the length of the challenge, since it follows from the CCA-security, that the attacker could not have learned anything in the first phase that would help him to solve the current challenge. Observe that CPA security does not seem to be sufficient, because the adversary may choose the challenges sent to the actor in the learning phase maliciously and the actor is returning the decryption. Thus, we are providing the adversary with a decryption oracle and this implies that we need a CCA secure encryption scheme.

C. Man-in-the-Middle Attacks and Denial of Service Attacks

We let the token sign the header in order to prevent man-in-the-middle attacks. In particular, we assume that the user running the protocol verifies that the timestamp stored in the header, the validity the public-key and the validity of the signature of the header. Denial of service attacks (DOS) are possible whenever the adversary prevents either the token or the actor to connect to the PKI. An obvious counter measurement is to let the token and the actor decide whether they trust the validity of the public-key. This might be reasonable in some scenarios and it provides the same level of security that is achieved nowadays. Further DOS attacks include turning the token off, destroying the token’s display, or preventing the delivery of the smart card. All these attacks cannot be prevented in the current systems either.

IV. CONTENT VERIFICATION

In this section we introduce content verification, where the basic idea is the verification of tokens, such as printed documents, with the help of digital signatures. Verifying digital signatures on printed documents is a challenging task for several reasons. Firstly, the document’s content must be human-readable, which prevents us of 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 have to apply a technique from computer vision, called optical character recognition (OCR). The issue, however, is that OCR has to be performed without any errors and from a practical point of view OCR is very unlikely to succeed error-free when reading a whole letter page with an unknown layout. Another issue is 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.

A. VeriDoc Machine-readable Document Format

To overcome the aforementioned problems and provide a practical solution that can be deployed in the real world, we developed a novel document format, called VeriDoc. This document format combines human and machine readable encodings with the idea being that the human can read the text while the HMD can read encoded information about the text, such as some layout information or a signature of the document’s content, using the machine readable encodings. We explain VeriDoc with the help of Figure 5. The document format consists of three components: a document header, a set of message blocks, and QR-Codes next to each of the message blocks. The header (cf. Figure 5(a)), which is encoded as a QR-Code, stores document information shown in Figure 6(a). In particular five variables are stored, where ver stores the version number of the content verification application, alg, the used signature scheme, sid, the signer’s identity, did, a document identifier, and nb, contains the number of blocks used in this document. The field layout, stores additional layout information of the document, and signature, contains the signature on the header itself. Each message block is printed along with a QR-Code that contains the corresponding signature and a block id (bid), which is given to a block according to its position (cf. Figure 5(b)).

Figure 5. The machine-readable VeriDoc document format. Information on the operation mode, layout, etc. is stored in the header QR-Code at the top of the document (a). Signatures are stored in individual QR-Codes printed next to each text block (b).

Figure 6. (a) Document header used in the content verification application and (b) the structure of the QR code printed/displayed next to each message block.

B. Our scheme

In this section we describe our content verification scheme. The scheme is based on standard cryptographic components such as unforgeable digital signature schemes $DS = (K_{ Sig}, \ Sig, V_f)$, and collision-resistant hash functions $H$. By sk we denote the private signing key and vk is the public verification key of the signer.
1) Signature Creation: In the following we describe the creating of signature and we refer to Algorithm 1 for a formal description. The input of the signing algorithm is a private key sk and a message \( m = m_1, \ldots, m_\ell \) consisting of \( \ell \) blocks. First, the algorithm computes the header token \( \text{TOKEN}_H \) of the document that stores the version of our protocol ver, the used algorithm \( \text{alg} \), the layout information \( \text{layout} \), a freshly chosen document id \( \text{did} \), the signer’s identity \( \text{sid} \), and a signature \( \sigma_H \) on these data. The second step is to create a signature for each message block. Each signature is computed over the current message block, the document id \( \text{did} \), and its position in the document bid.

Algorithm 1: Signing

input : \( m, \text{sid}, sk \)
output: Header token \( \text{TOKEN}_H \) and side tokens \( \text{TOKEN}_1, \ldots, \text{TOKEN}_\ell \)

Compute document header
set ver and alg;
set layout;
choose a random \( \text{did} \leftarrow \{0,1\}^n \);
parse \( m = m_1, \ldots, m_\ell \);
set \( \text{nb} := \ell \);
set \( h_H \leftarrow H(\text{ver}, \text{alg}, \text{sid}, \text{did}, \text{nb}, \text{layout}) \);
set \( \sigma_H \leftarrow \text{Sig}(sk, h_H) \);
set \( \text{TOKEN}_H \leftarrow \text{Encode}(\text{ver}, \text{alg}, \text{sid}, \text{did}, \text{nb}, \text{layout}, \sigma_H) \);

Compute QR codes for each message block
for \( i = 1, \ldots, \ell \) do
\[ h_i \leftarrow H(\text{did}, m_i, i) \]
\[ \sigma_i \leftarrow \text{Sig}(sk, h_i) \]
\[ \text{TOKEN}_i \leftarrow \text{Encode}(i, \sigma_i) \]
return \( \text{TOKEN}_H, \text{TOKEN}_1, \ldots, \text{TOKEN}_\ell \)

2) Content Verification Algorithm: Intuitively, the content verification algorithm checks the document’s validity by checking the validity of the signer’s public key\(^3\) the validity of the signed header, and then verifying each signed block separately. An algorithmic description of the verification process is given in Algorithm 2. The extraction of a message block, the corresponding signature, as well as the underlying computer vision techniques that are used, are simplified to \( \text{OCR}(\cdot) \) in this description. The computer vision techniques will be discussed in detail in Section VII-A.

To check the validity of the digital signature on a printed document, the verification algorithm first reads the authenticated document header \( \text{TOKEN}_H \) and extracts the version \( \text{ver} \), the used algorithm \( \text{alg} \), the signer’s identity \( \text{sid} \), the document identifier \( \text{did} \), the number of blocks \( \text{nb} \), the layout information \( \text{layout} \) and the signature \( \sigma_H \). The verification key \( \text{vk} \), that belongs to \( \text{sid} \) is obtained from a PKI. The algorithm outputs

Algorithm 2: Verify

input : Document D
output: Valid or Invalid

Verify the document header
\( (\text{ver}, \text{alg}, \text{vk}, \text{did}, \text{nb}, \text{layout}, \sigma_H) \leftarrow \text{Decode}(\text{TOKEN}_H) \);
set \( h_H \leftarrow H(\text{ver}, \text{alg}, \text{sid}, \text{did}, \text{nb}, \text{layout}) \);
set \( \text{vk} \leftarrow \text{PKI} (\text{sid}) \);
return 0 if \( \text{vk} \) is invalid;
return 0 if \( \text{Vf}(\text{vk}, h_H, \sigma_H) = 0 \);

Verify each message block
for \( i = 1, \ldots, \text{nb} \) do
\[ m_i, \text{TOKEN}_i \leftarrow \text{OCR}(b_i) \text{(see Section VII-A)} \]
\[ h_i \leftarrow H(\text{did}, m_i, i) \]
\[ i, \sigma_i \leftarrow \text{Decode}(\text{TOKEN}_i) \]
return 0 if \( \text{Vf}(\text{vk}, h_i, \sigma_i) = 0 \);

0 if \( \text{vk} \) is not valid and it also outputs 0 if the signature \( \sigma_H \) does not authenticate the document header. Afterwards, the verification algorithm checks the validity of each block. To do so, the algorithm first reads the message block along with its signature \( m_i, \text{TOKEN}_i \leftarrow \text{OCR}(b_i) \), it computes the hash value \( h_i \leftarrow H(\text{did}, m_i, i) \), extracts the signature from the corresponding QR code \( \sigma_i \leftarrow \text{Decode}(\text{TOKEN}_i) \) and outputs 0 if the signature is invalid, i.e., if \( \text{Vf}(\text{vk}, h_i, \sigma_i) = 0 \). If all checks are valid, then the verification algorithm outputs 1.

C. Threat Model

Our threat model follows the common notion of unforgeability, but is adapted to the PAM. The goal of an adversary in the PAM is to generate a new valid token, that verifies under some actor’s public key after obtaining a series of signed tokens for messages of his choice.

a) Query Phase: The adversary obtains as input all systems parameters and the public key of the honest actor. In addition, he may also send a polynomial number of adaptively chosen messages to the honest actor in order to obtain the corresponding signed tokens.

b) Output Phase: In the output phase the adversary outputs a new message and a token. The adversary wins the game if the message was not queried in the query phase and if the token verifies the authenticity of the message under the actor’s public key.

D. Security Analysis

We assume that the underlying signature scheme \( D = (\text{Kgsig}, \text{Sig}, \text{Vf}) \) that is used to generate the signed tokens is secure against existential forgery under an adaptive chosen message attack (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 \( \text{did} \in \{0,1\}^n \) is generated and the header also contains the number of blocks \( \text{nib} \). 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 numerated 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, we conclude that the resulting scheme is also existentially unforgeable under chosen message attacks.

V. TWO-FACTOR VERIFICATION

Using the content verification techniques described above, we introduce a new technique called two-factor 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 actors to use a technique known as two-factor authentication (sometimes called two-step verification) \[5\], 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\[7\]. In a similar vain we introduce the two-factor verification technique, that introduces a second step into the process of verifying retrieved content. Consider, for example, an actor, who requests his account balance during a online banking session. If the machine that is used, is untrusted and possibly even compromised, then the actor cannot verify the correctness of the returned balance. To overcome this problem we use our content verification technique described in [Section IV], 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, the account balance, and verify its correctness. An attacker, who wants to convince an actor of a false statement, would need to compromise the machine, that is used by the actor, and the HMD simultaneously, which is considerably harder to achieve in practice. Due to the simplicity of the two-factor verification technique, it could easily be integrated into many existing systems immediately.

VI. CONTENT HIDING

In this section we introduce the content hiding, where we establish a secure channel between a physical document and the HMD. The main motivation is to allow an actor to read confidential documents in the presence of eavesdroppers. For instance, a business employee could read confidential business records in a public space, e.g., while travelling, without being concerned about leaking any of the confidential information to nosy bystanders. Applications using this technique are not limited to paper-based documents or tablet computers. Consider a untrusted machine through which a actor might want to access some confidential data. Using the content hiding technique, he could display the information and use his HMD to access it, without leaking any information to the untrusted machine he is using.

To provide such privacy guarantees in the presence of eavesdroppers, we print the document (resp. display the data) in an encrypted format such that the actor can decrypt the ciphertext using his secret key and the HMD.

A. Our Scheme

The Ubic framework uses QR-Codes in combination with a hybrid encryption scheme \[6\] to generate these encrypted documents. A hybrid encryption scheme combines public- and private-key encryption techniques in order to obtain a public-key encryption scheme with short ciphertexts. The basic idea of such a system is to encrypt the message using a private-key encryption scheme and to encrypt the used key with a public-key encryption scheme. The document format that we are using is similar to one described in Figure 5 with the difference that the document only contains ciphertexts that are encoded in QR codes and no plaintext message blocks. The document header is shown in Figure 7(a) and the structure of each encrypted block is shown in (b). The header stores the application’s version \( \text{ver} \), the used public-key encryption scheme \( \text{pulbl} \), and the underlying private-key encryption scheme \( \text{priv} \). The field \( \text{key} \) stores the encrypted (freshly generated) private key. Optionally some layout information can be stored in the header as well. Each encoded block contains the block id \( \text{bid} \), as well as the encrypted data \( \text{data} \). The basic idea of such a system is to encrypt the message using a private-key encryption scheme and to encrypt the used key with a public-key encryption scheme. The document format that we are using is similar to one described in Figure 5 with the difference that the document only contains ciphertexts that are encoded in QR codes and no plaintext message blocks. The document header is shown in Figure 7(a) and the structure of each encrypted block is shown in (b). The header stores the application’s version \( \text{ver} \), the used public-key encryption scheme \( \text{pulbl} \), and the underlying private-key encryption scheme \( \text{priv} \). The field \( \text{key} \) stores the encrypted (freshly generated) private key. Optionally some layout information can be stored in the header as well. Each encoded block contains the block id \( \text{bid} \), as well as the encrypted data \( \text{data} \).

c) Encryption: To describe the encryption algorithm formally, we denote by \( \Pi_{\text{pke}} = (\text{Gen, Enc, Dec}) \) a public-key, by \( \Pi_{\text{priv}} = (\mathcal{G}, \mathcal{E}, \mathcal{D}) \) a private-key encryption scheme, and by \( \text{VE} = (\text{Encode, Decode}) \) a visual encoder. The following description of the encryption algorithm refers to Algorithm 3.

Figure 7. (a) Header used in the content hiding protocol and (b) header used for each block.
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 $TOKEN_H$ is created, which contains the framework header, the application’s version, information about which private-key encryption schemes were used, and the encrypted document key.

The actual body of the document $m$ is split into message chunks $m_1, \ldots, m_\ell$ and each chunk, together with its block id, is encrypted separately using the document key and is then encoded into a QR-Code $TOKEN_i$.

d) Decryption: The description of the decryption algorithm refers to the algorithm depicted in Algorithm 4. Upon receiving a document, the receiver reads the header QR-Code, obtains information about which schemes are used and retrieves the encrypted document key $k$. Using his secret key $dk$ the algorithm recovers the document key $k$ and it uses the key to decrypt the document body.

The advantages of representing the document as a sequence of QR codes are twofold. Firstly it allows the actor to only decrypt the part of the encrypted document body, that the actor is currently looking at, without requiring the user to scan the whole document first. Furthermore, the resulting documents are robust to damages. That is, even if a part of the encrypted document is broken or unreadable, we are still able to decrypt the remaining unimpaired 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 of them need to be displayed.

### B. Threat Model

In this scenario the adversaries goal is to learn information about the plaintext, given only the encrypted document. We model this security requirement through an indistinguishability game, which is similar to the standard CCA-security notion.

e) Learning Phase: The adversary communicates with an honest actor. The adversary can query the actor with messages that shall be encrypted, or with encrypted documents, that shall be decrypted. The actor generates the encrypted documents according to the scheme described in VI-A.

f) Challenge Phase: In the challenge phase, the adversary picks two messages of his choice and sends them to the actor. The actor picks one of the messages randomly and sends back the encrypted version of it. The attacker wins the game if he is able to decide which message was encrypted with a probability of at least $\frac{1}{2} + \epsilon(n)$, where $\epsilon$ is a non-negligible function and $n$ is the security parameter.

### C. Security Analysis

For our security analysis we assume that both $\Pi_{ple}$ and $\Pi_{priv}$ are CCA secure public/private key encryption schemes. Since both schemes are CCA secure, it is easy to see that the adversary does not learn any additional information about the encrypted message or any of the message blocks, hence our construction is secure w.r.t. out model.

### D. Achieving Non-Malleability

The system described above is malleable in the sense an adversary can simply remove blocks. This attack is possible because the message blocks are first split into chunks and then encrypted block-by-block. A simple way to prevent malleability attacks in general is to compute a message-authentication code (MAC) over all ciphertext blocks.

More precisely, the encryption algorithm is identical to the one described in Algorithm 3 with the difference that an additional key $k_M$ for the MAC is generated. The encryption algorithm then computes all ciphertexts $c_1, \ldots, c_\ell$ and stores the MAC on the message $M = c_1 || \ldots || c_\ell$ in the header as well. The decryption algorithm remains almost the same, except that in addition to recovering the document key $k$, it...
also recovers the key $k_M$, and once all ciphertexts $c_1, \ldots, c_\ell$ are read, it verifies the MAC on the message $M = c_1\| \ldots \| c_\ell$. It is well known that a CPA secure encryption scheme in combination with a MAC yields a CCA secure encryption scheme [7].

However, it should be noted that the MAC can not be computed if some blocks are destroyed. The ciphertext contained in the remaining blocks can still be decrypted, but the MAC cannot be verified anymore.

E. Extending Content Hiding to Support Fine-Grained Access Control

With our basic content hiding scheme we can encrypt documents for a certain recipient. However, in many companies or organizations it is more desirable to encrypt documents, such that a certain class of employees, which have some well-defined clearances, can read certain encrypted documents. We extend our scheme to support such fine grained read access structures, such that only actors with certain clearances can read certain document. We achieve this by replacing the public-key encryption scheme by a predicate encryption scheme [8]. 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 $\Sigma$ is the universe of all possible attributes. The encryption algorithm outputs a ciphertext that can only 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. A comprehensive description of the predicate encryption scheme we deployed can be found in Appendix A.

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 we denote by $Prof$, $Emp$, and $Stud$ the corresponding attributes. Every member of a group will be equipped with a secret key $sk_f$ such that $f$ is either the predicate $mayAccProf$, $mayAccEmp$, or $mayAccStud$. 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 says that a file $file$ with is encrypted using the attribute $Prof$, cannot be decrypted by a student equipped with $sk_{mayAccStud}$ and the student also cannot tell with which attribute it is encrypted except for 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.

g) 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 person encrypting the message chooses in addition an attribute $I \in \Sigma$ that specifies which user can decrypt the message. Formally, our encryption algorithm is almost the same as described in Algorithm 3 but the public-key encryption step is replaced with the following $c \leftarrow PrEnc(mpk, I, m)$ (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 $PrDec(sk_f, c)$ and the actor can only decrypt if and only if $f(I) = 1$.

VII. Prototype Implementation

We implemented a fully functional prototype of our Ubic framework on a laptop computer and a working implementation on Google Glass will be made available as open source upon publication. Amongst other sensors and most important for the purposes of our framework, Google Glass features a 640x360 optical head-mounted display as well as an egocentric camera with a resolution of 1280x720 pixels. The currently available developer version only features an embedded microcontroller with 1.2 GHz and 1GB of memory. For this reason we opted to use computer vision techniques with low computational complexity.

To provide a better understanding of how our system works, we provide a demo video under the following URL: https://www.dropbox.com/s/0awzy74d2jydaqlubicrypt.mp4. The link to the video is an anonymous link to some dropbox space and does not provide any information about who the authors are. The person shown at the beginning of video is a random student and has no connection to the authors. Hence this video does not compromise the anonymity requirements of the submission.

It shows a little demo presentation of the identification, the content verification and content hiding functionality. The black screen which displays the information is the screen that is displayed on the Google Glass display. It covers the whole display but only a small part of the users view. It should be noted that the video stutters due to the recording software. Our implementation itself runs smoothly and does not stutter.

A. The VeriDoc Interface

An overview of the workflow in the content verification scenario is shown in Figure 8. Throughout the interaction we have taken care that the process remains transparent to the user. First, the user is asked to scan the header QR code, that is the starting point for any type of interaction that we propose here. As described above, the header contains rich information about the document structure, encryption types and functionalities that this document supports. In this case, it allows the verification of the content and therefore continues by asking the user to scan the first text block using the front-facing camera of Google Glass. The brackets displayed on the head-mounted display help the user to position the camera properly. The exact alignment as well as the content extraction is further facilitated by a computer vision subsystem as described below. After each scanned block, its extracted content on the left side is verified with the signature encoded in the QR-Code on the right side. The user is informed about
the validity of each text block and once all blocks of a given document are scanned a final prompt terminates the interaction and informs the user if the document as a whole was verified or not.

1) 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 rest of the computer vision processing chain (see Figure 9). 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 $M$ in order to robustly identify corners:

$$A = g(\sigma_2) \ast \begin{pmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{pmatrix}$$ (1)

$$M = \det(A) - \kappa \text{trace}(A)^2$$ (2)

where $I_x$ and $I_y$ are the spatial image derivatives in x and y direction, $\sigma_D$ smoothing of the image with the detection scale and $\sigma_I$ smoothing the response with the integration scale and $\kappa = 0.04$ according to best practice. Intuitively, the pre-smoothing with $\sigma_D$ eliminates noise and allows detection of corners at a desired scale [10] while the smoothing $\sigma_I$ 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 [11].

The second image on 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 $H \in \mathbb{R}^{3 \times 3}$ 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

$$p' = Hp$$ (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 associated operation of homography estimation and unwarping are part of OpenCV, a standard computer vision toolbox [12].

The image on the upper right shows the content after unwarping and cropping. Utilizing the information contained in the header we now split the content area into text and the text area is further processed using the tesseract OCR system.

B. Cryptographic Primitives and Application Specific Details

Our framework uses the RSA-OAEP public-key encryption scheme with 2048 bit long keys. SHA-1 is used as the collision-resistant hash function and we use SHA-1 in combination with RSA to sign messages. AES-256 is used as the private-key encryption scheme. The implementation of our predicate encryption scheme uses the jPBC library [11].

QR-Codes commonly expect alphanumeric character sequences, but encryption and signature algorithms return byte arrays. Therefore, we use base64-encoding to encode the byte arrays into character sequences before storing them in the header. Currently, all applications use these cryptographic primitives. In the following we provide a short overview of the application specific settings:

1) Identification: We used QR-Codes with a medium error correction to encode the encrypted six digits long decimal challenge with the corresponding side information. An actor, upon receiving such a QR-Code from the token, verifies the validity of the token’s public key and the signature of the challenge. The challenge is rejected if either one of these checks fails, or the the timestamp contained in the challenge is older than 10 minutes.

2) Content Verification: Depending on the amount of data, that needs to be stored in the document header, the error-
Figure 9. 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 unwraps and extracts the content of that block.

correction level is chosen. The layout information that is encoded in the document header is the aspect-ratio and the text-to-QR-Code-ratio for each block.

3) Content Hiding: In the content hiding scenario we use QR-Codes with a high error-correction to provide more robustness. We use a chunk size of 350 when splitting the message into blocks. We consider this size to be a good trade-off between the readability of each QR-Code and the number of codes that are needed to encrypt the text. The optional layout information, when set, indicates how many QR-Codes there are per line.

VIII. RELATED WORK

In [14] a broad range of different AR applications are examined. There, AR systems are defined as systems, that merge real and digital content interactively in real time. This definition does not limit itself to applications based on visual channels, but also covers, for instance, auditory channels. More recent work mainly focuses on providing fine-grained access controls to sensor feeds, such as video and audio. In [15], Jana et al. consider the scenario where a trusted device, e.g. a mobile phone, runs untrusted third-party applications, that require access to the device’s perceptual functionalities, e.g. its camera. Roughly speaking, they propose a framework, where applications only obtain a filtered camera output. The filters are chosen according to the application’s permissions. In [16] the notion of recognizers is introduced. Rather than passing a filtered sensor feed to the application, they provide a set of recognizers that fulfil the most common tasks, such as face detection or recognition. Applications then directly request the output of the recognizer, rather than executing these algorithms on the sensor stream by themselves. This line of research mainly concentrates on enforcing privacy on sensor feeds.

Another line of research concentrates on establishing trust between devices based on the visual channel [17], [18], [19]. In [19] the visual channel is used for demonstrative authentication. That is, a user identifies a device he wants to communicate with, with his own device by scanning a QR-code. For example, an access point can have a QR-code printed onto it and a user can use his camera-enabled device to authenticate that access point. Their approach has the advantage of being transparent for the user who points at the device he wants to authenticate.

In contrast to this work, rather than concentrating on one specific application, we concentrate on providing a more general-purpose framework. We introduce a new attacker model, that takes the physical channel into consideration and allows us to provide a more precise security analysis of applications that use this kind of channel. Our prototype implementation covers a broad range of important applications and it is designed to be easily extendable. To the best of our knowledge we are the first to provide a framework for using HMDs for privacy and security purposes.

IX. CONCLUSION AND FUTURE WORK

We have presented Ubic, a framework that makes an important step towards a tighter integration of digital cryptography in real-world applications. Using a new attacker model, we were able to model more realistic attackers, that have access to the physical channel between communicating parties. Ubic uses HMDs in combination with established cryptographic primitives and resource friendly computer vision techniques, to provide the user with more security and privacy guarantees in a wide range of common real-world applications. We introduced a new document format, which allows for robust document tracking, text-recognition, and the verification of its content. It provides a superior alternative to hand-written signature, since it has a well-defined verification algorithm. In addition, it allows companies to destroy the physical document and only store digital version thereof.

Based on these content verification techniques we introduced the two-factor verification technique, which allows a user to verify content that was obtained or is viewed through a untrusted device. Due to its simplicity it could be integrated into various existing systems immediately.

Exploiting the new possibilities opened up by HMDs, we showed how a user can read confidential document in the
presence of eavesdroppers. For example, this allows business people to inspect their confidential records while travelling without being worried about any nosy bystanders. We extended this basic idea with the help of predicate encryption schemes to be able to create documents with fine-grained read access policies, where a document is encrypted in such a way, that only people with certain clearances are able to read its content. This might be interesting for paper-based offices, where different employees have different clearances.

Furthermore, we showed how to use HMDs to identify users securely and conveniently in front of tokens, such as doors or ATMs.

Even though our prototype framework covers a broad range of different applications, it still work to be done. Many more applications can be thought of, and integrated into Ubic. For instance, business cards could have the owner’s public key printed onto them as QR-Codes. This would provide a convenient solution for a trusted key distribution. Apart from extending Ubic with new applications based on the visual channel, we also envision our framework to be extended with applications, that use different channels such as the auditory channel. The new versions of Google Glass will be shipped with headphones and one could, for instance, think of encrypted audio broadcasts.

We hope the further developments and improvements will help us to narrow the gap between real-world applications and digital cryptographic primitives even further.

REFERENCES

[1] R. Beck. (2013, Oct.) Legal dispute about scanned business documents. [Online]. Available: http://www.heise.de/az/meldung/ Eingescannte-Rechnungen-und-Vertraege-werden-zum-Streitfall-197604.html

[2] International Organization for Standardization, “Information technology — automatic identification and data capture techniques — qr code 2005 bar code symbology specification,” 2006.

[3] S. B. Wicker, Reed-Solomon Codes and Their Applications. Piscataway, NJ, USA: IEEE Press, 1994.

[4] M. Kumar, T. Garfinkel, D. Boneh, and T. Winograd, “Reducing shoulder-surfing by using gaze-based password entry,” in Proceedings of the 3rd Symposium on Usable Privacy and Security, ser. SOUPS ’07. ACM, 2007, pp. 13–19.

[5] R. M. Van Rijswijk and J. Van Dijk, “Tiqr: A novel take on two-factor authentication,” in Proceedings of the 25th International Conference on Large Installation System Administration, ser. LISA’11. USENIX Association, 2011, pp. 7–7.

[6] R. Cramer and V. Shoup, “Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack,” SIAM J. Comput., vol. 33, no. 1, pp. 167–226, Jan. 2004.

[7] D. Dolev, C. Dwork, and M. Naor, “Non-malleable cryptography,” SIAM Journal on Computing, vol. 30, no. 2, pp. 391–437, 2000.

[8] J. Katz, A. Sahai, and B. Waters, “Predicate encryption supporting disjunctions, polynomial equations, and inner products,” in EUROCRYPT, ser. Lecture Notes in Computer Science, N. P. Smart, Ed., vol. 4965. Springer, 2008, pp. 146–162.

[9] C. Harris and M. Stephens, “A combined corner and edge detector,” in Proceedings of the 4th Alvey Vision Conference, 1988, pp. 147–151.

[10] T. Lindeberg, Scale-Space Theory in Computer Vision. Norwell, MA, USA: Kluwer Academic Publishers, 1994.

[11] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 27, no. 10, pp. 1615–1630, 2005.

[12] G. Bradski, “Open source computer vision library (opencv),” 2000. [Online]. Available: http://opencv.org/

[13] R. Smith, “An overview of the tesseract ocr engine,” in Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02, ser. ICAR '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 629–633. [Online]. Available: http://dl.acm.org/citation.cfm?id=1304596.1304846

[14] R. Azuma, “A survey of augmented reality,” 1997.

[15] S. Jana, A. Narayanan, and V. Shmatikov, “A scanner darkly: Protecting user privacy from perceptual applications,” in IEEE Symposium on Security and Privacy. IEEE Computer Society, 2013, pp. 349–363.

[16] S. Jana, D. Molnar, A. Moshchuk, A. Dunn, B. Livshits, H. J. Wang, and E. Ofek, “Enabling Fine-Grained Permissions for Augmented Reality Applications With Recognizers,” in 22nd USENIX Security Symposium (USENIX Security ’13), Washington DC, August 2013.

[17] G. Starnberger, L. Frohhofer, and K. M. Goeschka, “Qr-tan: Secure mobile transaction authentication,” 2012 Seventh International Conference on Availability, Reliability and Security, vol. 0, pp. 578–583, 2009.

[18] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan, “Secure device pairing based on a visual channel,” in IN 2006 IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 2006, pp. 306–313.

[19] J. M. McCune, A. Perrig, and M. K. Reiter, “Seeing-is-believing: Using camera phones for human-verifiable authentication,” in In IEEE Symposium on Security and Privacy, 2005, pp. 110–124.

[20] D. M. Freeman, “Converting pairing-based cryptosystems from composite-order groups to prime-order groups.” in EUROCRYPT, ser. Lecture Notes in Computer Science, H. Gilbert, Ed., vol. 6110. Springer, 2010, pp. 44–61.

APPENDIX

For completeness, we recall the predicate encryption scheme due to Katz, Sahai, and Waters [8]. However, due to efficiency reasons, we did not implement the scheme in composite order groups but adapted the transformation to prime order groups as suggested by Freeman [20].

A. Definition

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

A predicate encryption scheme \( \Pi_{PE} \) is correct if and only if, for all \( \lambda \), all key pairs \((\text{mpk}, \text{psk}) \leftarrow \text{PrGen}(1^\lambda) \), all predicates \( f \in \mathcal{F} \), all secret keys \( \text{sk}_f \leftarrow \text{PrKGen}(\text{psk}, f) \), and all attributes \( I \in \Sigma \) we have that (i) if \( f(I) = 1 \) then \( \text{PrDec}(\text{sk}_f, \text{PrEnc}(\text{mpk}, I, m)) = m \) and (ii) if \( f(I) = 0 \) then \( \text{PrDec}(\text{sk}_f, \text{PrEnc}(\text{mpk}, I, m)) = \perp \) 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 $G$ be an order-$N$ cyclic group and $e : G \times G \rightarrow G_T$ 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 $(g) = e$ then $e(g, g) \neq 1$. Then, by the chinese remainder theorem, $G = G_p \times G_q \times G_r$ where $G_s$ with $s \in \{p, q, r\}$ are the $s$-order subgroups of $G$. Moreover, given a generator $g$ for $G$, $(g^p_q) = G_r$, $(g^p_r) = G_q$, and $(g^r_p) = G_p$. Another insight is the following, given for instance $a \in G_p$ and $b \in G_q$, we have $e(a, b) = e((g^p_q)^a, (g^p_r)^b) = e(g^r_p, g^p_q)^{apb}$, i.e., a pairing of elements from different subgroups cancels out. Finally, let $G$ be an algorithm that takes as input a security parameter $\lambda$ and outputs a description $(p, q, r, G, G_T, e)$. We describe the algorithms PrGen, PrKGen, PrEnc, and PrDec in the sequel.

a) Algorithm PoGen($\lambda$, $n$) and PrGen($\lambda$, $n$): First, the algorithm runs $G(\lambda)$ to obtain $(p, q, r, G, G_T, e)$ with $G = G_p \times G_q \times G_r$. Then, it computes $g_p$, $g_q$, and $g_r$ as generators of $G_p$, $G_q$, and $G_r$, respectively. The algorithm selects $R_0 \in G_r$, $R_1 = R_2 = 1 \in G_r$ and $h_1, h_2 \in G_p$ uniformly at random for $1 \leq i \leq n$. $(N = pqr, G, G_T, e)$ constitutes the public parameters. The public key for the predicate-only encryption scheme is

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

and the master secret key is

$$\text{osk} = (p, q, r, g_q, \{h_{1,i}, h_{2,i}\}_{i=1}^n).$$

For the predicate encryption with messages, the algorithm additionally chooses $\gamma \in \mathbb{Z}_N$ and $h \in G_p$ at random. The public key is

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

and the master secret key is

$$\text{psk} = (p, q, r, g_q, h^{-\gamma}, \{h_{1,i}, h_{2,i}\}_{i=1}^n).$$

b) Algorithm PoKGen(osk, $\vec{v}$) and 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 $R_5 \in G_r$, random $f_1, f_2 \in G_q$, and random $Q_6 \in G_q$. For the predicate-only encryption scheme it outputs a secret key

$$\text{osk}_{\vec{v}} = \left(\begin{array}{l}
K_0 = R_5 \cdot Q_6 \cdot \prod_{i=1}^n h_{1,i}^{r_{1,i}} \cdot h_{2,i}^{r_{2,i}} ,
\{K_{i,1} = g_p^{r_{1,i}} \cdot g_{1,i} \cdot v_i, K_{i,2} = g_p^{r_{2,i}} \cdot g_{2,i} \cdot v_i\}_{i=1}^n
\end{array}\right).$$

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

$$K_0 = R_5 \cdot Q_6 \cdot h^{-\gamma} \cdot \prod_{i=1}^n h_{1,i}^{r_{1,i}} \cdot h_{2,i}^{r_{2,i}}.$$
For the predicate encryption scheme with messages, we have \( sk_\vec{v} = (K_0, \{ K_{1,i}, K_{2,i} \}_{i=0}^n) \) and a ciphertext \( C = (C', C_0, \{ C_{1,i}, C_{2,i} \}_{i=0}^n) \). Then

\[
C' \cdot e(C_0, K_0) \cdot \prod_{i=0}^n e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i})
\]

\[
= m \cdot P^s \cdot e(g_p^s, R_5, Q_6 \cdot h^{-\gamma} \cdot \prod_{i=1}^n h_{1,i}^{-r_1,i} \cdot h_{2,i}^{-r_2,i})
\]

\[
\cdot \prod_{i=1}^n e(H_{1,i}^s, Q^{\alpha \cdot x_i} \cdot R_{3,i}, g_{p}^{r_1,i} \cdot g_{q}^{r_2,i})
\]

\[
\cdot e(H_{2,i}^s, Q^{\beta \cdot x_i} \cdot R_{4,i}, g_{p}^{r_2,i} \cdot g_{q}^{r_2,i})
\]

\[
= m \cdot e(g_p, h)^{s \gamma} \cdot e(g_p, h)^{s \gamma}
\]

\[
\cdot \prod_{i=1}^n e(g_p, h_{1,i})^{s r_1,i} \cdot e(g_p, h_{2,i})^{s r_2,i}
\]

\[
\cdot \prod_{i=1}^n e(h_{1,i}, g_p)^{s r_1,i} \cdot e(g_q, g_q)^{\alpha x_i \cdot f_1 \cdot v_i}
\]

\[
\cdot e(h_{2,i}, g_p)^{s r_2,i} \cdot e(g_q, g_q)^{\beta x_i \cdot f_2 \cdot v_i}
\]

\[
= m \cdot \prod_{i=1}^n e(g_q, g_q)^{s (\alpha f_1 + \beta f_2) x_i \cdot v_i}
\]

\[
= m \cdot e(g_q, g_q)^{s (\alpha f_1 + \beta f_2) \cdot \langle \vec{x}, \vec{v} \rangle}
\]

The second factor in the last line cancels out only if \( \langle \vec{x}, \vec{v} \rangle = 0 \), as expected.