6-7 Two Approaches to Analyzing the Security of Cryptographic Protocols

Maki YOSHIDA

Cryptographic protocols are designed to achieve various security goals, such as data confidentiality and entity authentication. However, critical vulnerabilities in protocol specifications have been frequently reported. Thus, one of the most important issues is the development of methods of analyzing the security of cryptographic protocols. This report presents our main contributions on protocol analysis in the two approaches, known as the manual (or the cryptographic) approach and the formal (or the symbolic) approach.

1 Introduction

Cryptographic protocols are designed to achieve various security goals, such as data confidentiality and entity authentication, even when the communication takes place over unsecure networks. In recent years, critical vulnerabilities in protocol specifications have been frequently reported [1]–[3]. For example, on 14th of October 2014, Google researchers announced a new vulnerability in SSLv3 (Secure Socket Layer version 3.0) that allows man-in-the-middle attackers to obtain clear text data, aka the "POODLE" attack [1]. For early detection of such vulnerabilities in protocol specifications, the development of analysis methods is in urgent need.

There exist two approaches for analyzing the security of cryptographic protocols, known as the manual (or the cryptographic) approach and the formal (or the symbolic) approach. In both approaches, it is assumed that the used cryptographic primitives are secure. The central features of the manual approach are detailed models for various types of cryptographic protocols and attackers in the cryptographic model. From these features, the resulting security proofs offer powerful security guarantees. A serious drawback of this approach is that proofs become more difficult, tedious, and error-prone if we try to analyze more complex protocols against powerful attackers. In contrast, the formal approach that abstracts the details about the protocol specifications and environments in the symbolic model leads to considerably simpler proofs, and can benefit from machine support. However, it has been proved that no general formal method that analyzes the security for *all* possible cryptographic protocols exists. These advantages and drawbacks make it important to devise various methods through both approaches for successful analysis.

This report presents our main contributions on protocol analysis in the two approaches — proof of powerful security [4] and abstraction for automated analysis [5]. In [4], through the manual approach, we derived a lower bound on the communication complexity of an *arbitrary* cryptographic protocol that realizes non-interactive *secure* multi-party computation. This bound clarifies the limit of efficiency, and further provides a criterion for security analysis because any secure protocol cannot exceed this bound. If the communication complexity exceeds the bound, we can conclude that the protocol is vulnerable. Then, we presented an efficient construction of secure protocols, which shows an upper bound on the communication complexity. In [5], we improved the previous formal approach and presented an abstraction method that guarantees exhaustive detection of attacks against a specific class of cryptographic protocols that use public-key cryptosystems. Although being applicable to a limited class of cryptographic protocols, the global share of public keys such as the public key infrastructure is allowed. This contributes to automated analysis for practical environments.

The rest of this report is organized as follows. Section **2** shows the derived lower and upper bounds on the communication complexity of cryptographic protocols for non-interactive secure multiparty computation in [4]. Section **3** presents an overview of the improved formal approach and abstraction method in [5]. Finally, Section **4** concludes this report.

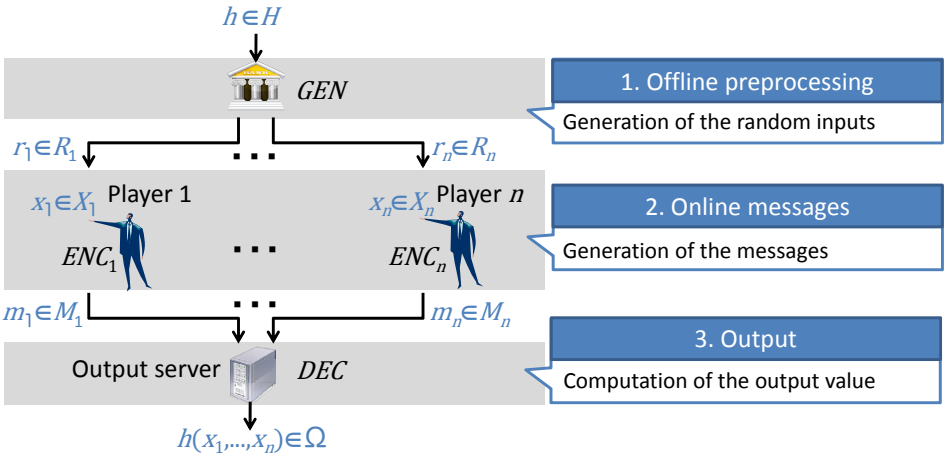

Fig. 1 An NIMPC protocol Π for H

2 Cryptographic analysis

Secure multi-party computation (MPC) aims to enable multiple players to co-operatively compute various functions in the presence of adversaries. MPC was first introduced by Yao [6] and because of its importance in cryptography, there have been presented many variants so far [7]–[9]. We consider non-interactive MPC (NIMPC) against honest-but-curious adversaries in the information-theoretic setting, which was introduced by Beimel et al. in [9].

2.1 Non-interactive secure multi-party computation

We recall the notations and definitions of NIMPC introduced in [9]. Let n be the number of players and $[n] = \{1, \dots, n\}$. Let $X_1, \dots, X_n, R_1, \dots, R_n, M_1, \dots, M_n$, and Ω be finite domains. Let H be a family of functions $h: X_1 \times \cdots X_n \to \Omega$. An NIMPC protocol for H is a triplet $\Pi = (GEN, \{ENC_i\}_{i\in[n]}, DEC)$ where

- a randomness generation function $GEN: H \to R_1 \times \cdots \times R_n$ given a description of a function $h \in H$ generates n correlated random inputs $r_i \in R_i$ with $i \in [n]$,
- a local encoding function $ENC_i: X_i \times R_i \to M_i$ takes an input $x_i \in X_i$ and the correlated random input $r_i \in R_i$ and outputs a message $m_i \in M_i$,
- a decoding algorithm $DEC: M_1 \times \cdots \times M_n \to \Omega$ reconstructs $h(x_1, \cdots, x_n) \in \Omega$ from the n messages $m_i \in M_i$ with $i \in [n]$.

An NIMPC protocol is also described in the language of protocols. Such a protocol involves n players, each holding an input $x_i \in X_i$, and an external "output server" with no input (see Fig. 1). For an NIMPC protocol $\Pi = (GEN, \{ENC_i\}_{i\in[n]}, DEC)$ for a class of functions H, let $P(\Pi)$ denote the protocol that may have an additional input, a function $h \in H$, and proceeds as follow.

- **Offline preprocessing:** Each player $i \in [n]$ receives the random input $r_i \in R_i$ where $(r_1, \dots, r_n) \leftarrow GEN(h)$.
- **Online messages:** On the random input $r_i \in R_i$, each player $i \in [n]$ sends the message $m_i = ENC_i(x_i, r_i) \in M_i$ to the output server.
- **Output:** The output server computes and outputs $DEC(m_1, \dots, m_n) \in \Omega$.

The communication complexity is evaluated by the summation of $\log_2|R_1|, \dots, \log_2|R_n|$, and that of $\log_2|M_1|, \dots, \log_2|M_n|$.

The requirements for an NIMPC protocol $\Pi = (GEN, \{ENC_i\}_{i\in[n]}, DEC)$ are defined as follows.

- **Correctness:** We say that Π is correct if the output server outputs the value $h(x_1, \dots, x_n)$ with probability 1. Specifically, for any $h \in H$, $x_i \in X_i$ with $i \in [n]$, and $(r_1, \dots, r_n) \leftarrow GEN(h)$, $DEC(ENC_1(x_1, r_1), \dots, ENC_n(x_n, r_n)) = h(x_1, \dots, x_n)$.
- **Robustness:** For a subset $T \subseteq [n]$, we say that Π is T-robust if the corrupted players T and the output server can simulate their view of the protocol (i.e., the random inputs $\{r_i\}_{i\in T}$ and the messages $\{m_i\}_{i\in[n]\setminus T}$) given oracle access to the function h restricted by the other inputs $\{x_i\}_{i\in[n]\setminus T}$ (see Fig. 2). We say that Π is fully robust (or simply refer to Π as an NIMPC protocol for H) if Π is T-robust for every $T \subseteq [n]$ of size $|T| \leq n$.

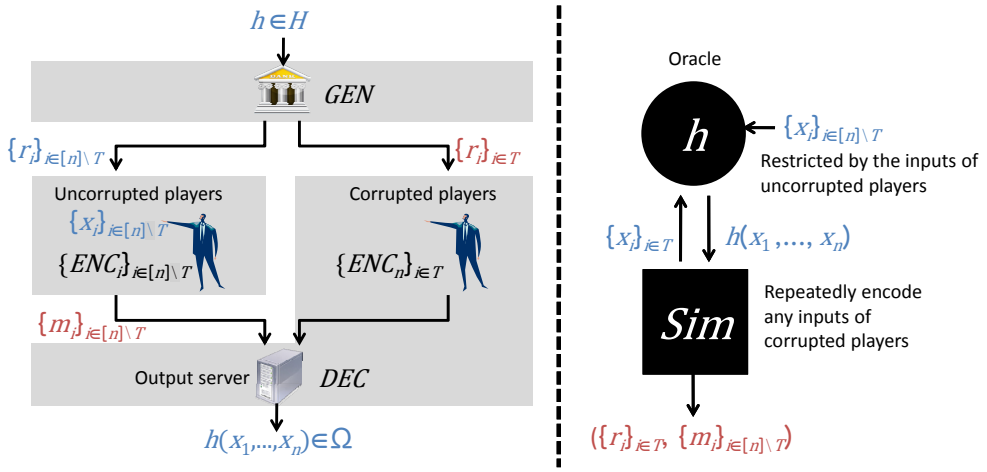

Fig. 2 *T*-robustness of NIMPC protocol Π for *H*

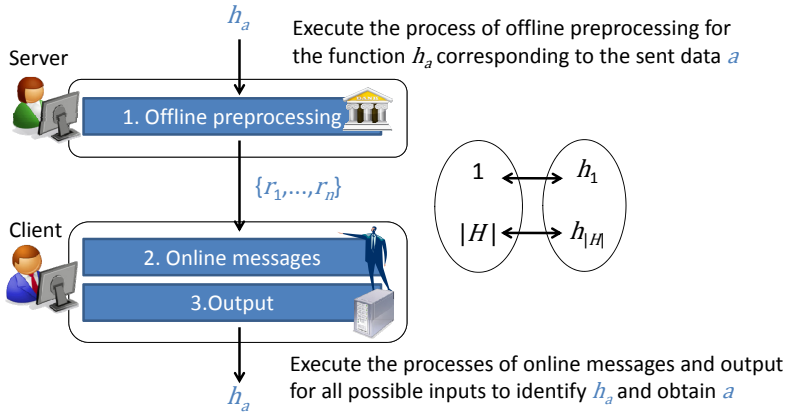

Fig. 3 Data transmission using an NIMPC protocol

2.2 Lower and upper bounds on the communication complexity and its application to security analysis

We derived a lower bound on the communication complexity of every fully robust NIMPC protocol Π for an arbitrary class of functions H. If the communication complexity of a protocol exceeds the bound, then we can see that the protocol is not fully robust and leaks some information on the function or inputs. The communication complexity is bounded by the size of target class. More specifically, the communication complexity cannot be smaller than the logarithm of the size of the target class $\log_2|H|$. Thus, we have a larger communication complexity if we allow a larger number of functions to be evaluated. Our technique for deriving lower bounds is quite simple and useful for approximating the amount of communication. For the target class of functions H, we first assume the existence of a *correct* NIMPC protocol $P(\Pi)$ with some communication complexity C and show a method for a server to send data to a client by encoding data $a \in \{1,\ldots,|H|\}$ into a function $h \in H$ and evaluating the function (see Fig. 3). Thus, the communication complexity C is bounded by the description length of h. If the assumed communication complexity is smaller than the logarithm of the size of the target class, i.e., $C < \log_2|H|$, then the contradiction is implied. Thus, it holds that $C \geq \log_2|H|$, and the communication complexity is lower bounded by $\log_2|H|$. We note that this lower bound is common to both information theoretically secure protocols and computationally secure ones because the proof only depend on the correctness.

Table 1 shows the communication complexity for two major classes of functions (the arbitrary functions and the indicator functions). An indicator function is the function defined to be identically one on a specific input, and is zero elsewhere. An arbitrary function is represented by the sum of indicator functions. For the class of indicator functions whose number equals to that of possible inputs $|X|$, the communication complexity is lower bounded by the input length $\log_2|X|$. For the class of arbitrary functions with

Table 1 The communication complexity of n-player NIMPC protocols for two families of functions $h: X_1 \times \cdots X_n \to \Omega$ where $X = X_1 \times \cdots \times X_n$ and $d = \max_{i \in [n]} |X_i|$.

	All functions (m-bit output)	The indicator functions (1-bit output)
Derived lower bound	$\|X\| \cdot m$	$\log_2 \|X\|$
Previous protocols in [9]	$\|X\| \cdot m \cdot d^2 \cdot n$	$d^2 \cdot n$
Proposed protocols	$\|X\| \cdot m \cdot \lceil \log_2 d \rceil^2 \cdot n$	$\lceil \log_2 d \rceil^2 \cdot n$

m-bit output, the lower bound is $|X| \cdot m$. However, there is an exponential gap between the derived lower bound and the previous construction in [9].

We then reduced the gap between the lower and upper bounds to quadratic in the input length by presenting a more efficient construction of an NIMPC protocol for the indicator functions in [4].

3 Symbolic analysis

We improve the formal approach proposed by Canetti and Herzog in [11], called the universally composable symbolic analysis, and present a method to abstract cryptographic protocols that share keys via Public-Key Infrastructure (PKI).

3.1 Improving the universally composable symbolic analysis approach

In [11], Canetti and Herzog have proposed the following approach.

- Using the universal composition framework (UC): The UC framework [10] provides a general way for specifying the security requirements of cryptographic tasks and asserting whether a given protocol realizes the specification. A salient property of this framework is that it provides strong composability guarantees: a protocol that realizes the specification in isolation continues to realize the specification regardless of the activity in the rest of the network. Thus, it is enough to abstract a single session of the isolated protocol.
- Proving the computational soundness of the abstracted (symbolic) model: The computational soundness here means that if the adversary in the UC setting can do nothing, then the abstracted, symbolic adversary cannot also do (except with negligible probability). This allows us to apply an automated symbolic method to security analysis.

The UC framework used by the previous work in [11], [12] is UC with Joint State (JUC) [13]. The JUC framework provides a means to deduce the security of the multi-session case from the security of a single session, even when some joint state is used by each protocol (e.g., the protocol-private keys) as shown in the right side of Fig. 4. However, this framework is not suitable for real-world protocols in the Internet where keys are used by multiple protocols such as various versions of TLS.

In this work, to consider an execution of protocols in a setting involving a global setup (e.g., PKI) where keys are shared among multi-sessions of an arbitrary protocol, we use Externalized UC (EUC) [14] that provides the composition guarantee even when the same computational entity is used as a subroutine within multiple protocols as shown in the left side of Fig. 4.

To present the computationally sound abstraction in the EUC framework, we first define syntax and semantics of target protocols, called simple protocols, in both EUC framework (concrete model) and symbolic model (Fig. 5) We then define the *trace* of an execution of a simple protocol within the EUC framework. The trace provides a global view of the execution, consisting of a sequence of

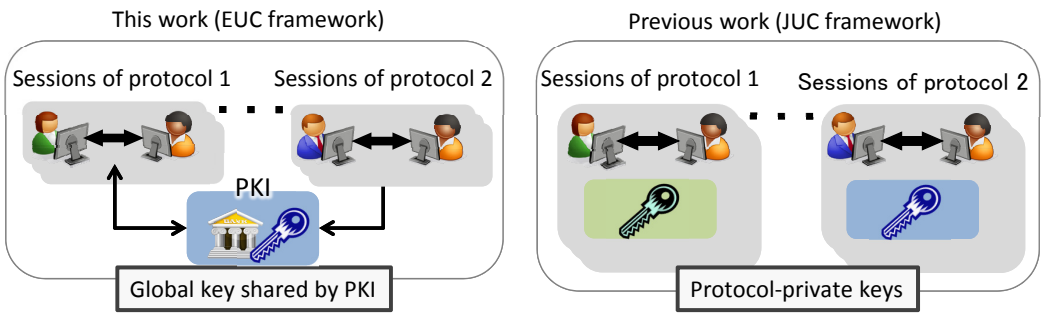

Fig. 4 Differences of the UC framework between this work and the previous work

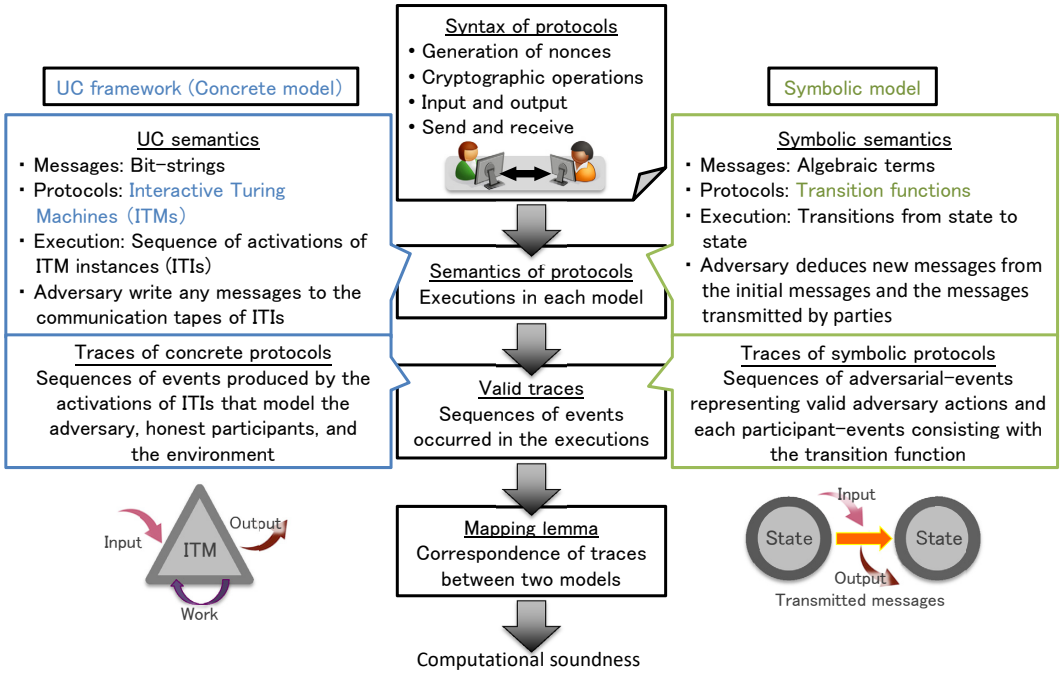

Fig. 5 An abstraction method to guarantee the computational soundness of a symbolic model

input, outputs, messages, and local variables (represented in bit-strings). It also contains the participants' calls to the shared functionality, thus capturing the global use of keys. In a similar way, we define the trace of an execution of a symbolic protocol within the symbolic model. Again, the trace represents a global view of the (now symbolic) execution. Here, the trace consists of a sequence of expressions from the underlying symbolic algebra. Next, we define a trace mapping which translates a trace of a concrete simple protocol into a symbolic trace. Finally, we show that this mapping provides soundness to trace properties in the symbolic protocol. That is, the trace mapping almost always (except for negligible probability) translates a trace of a concrete simple protocol to a valid trace of the corresponding symbolic protocol. The term "valid" here means that the trace could have been produced by the symbolic protocol. From this mapping lemma, if there is no attack in any symbolic trace, then we conclude the original concrete protocol is secure.

3.2 Overview of our computationally sound symbolic model

The central part in our abstraction is the mapping lemma that establishes a correspondence between executions of concrete simple protocols and executions of the corresponding symbolic protocols.

In the symbolic model that we use for abstracting protocols, messages are represented as compound elements in some symbolic algebra. That is, each compound element represents a "parse tree," or a sequence of operations needed to obtain the composite symbol from atomic symbols. The atomic symbols are used to represent primitive structures such as party identifiers, messages, random nonces, cryptographic operations, and their keys denoted by $ID, m, r, Enc, Dec, ek, dk, Sig, Ver, sk, vk$, and so on. The compound elements of the algebra (those messages produced by the operations with keys) represent those messages that encrypt or sign primitive messages such as $Enc(ek, m), Sig(sk, m)$.

A symbolic trace is a sequence of events, and a trace is valid for a protocol if the messages delivered by the adversary to the participants are consistent with the adversary's computations, and the messages sent by participants are consistent with the messages received and the protocol. The symbolic adversary can deduce new messages from the initial messages and the messages generated by parties running the protocol. The adversary operations are extremely limited. Specifically, the adversary cannot perform any operations other than the symbolic ones such as concatenate messages, decompose elements of a message, encrypt a message with a given public key, or decrypt a given symbolic ciphertext if the corresponding secret key is corrupted.

The translation of a concrete message to a symbolic one requires knowledge of events that occur in the trace. The previous work based on the JUC framework [11][12]

concentrates on a restricted class of protocols that use no cryptographic primitives other than JUC-secure public-key encryption. In conjunction with the treatment of joint state, the use of public-key encryption is modeled as interaction with an ideal functionality. Thus, by observing all calls to the ideal functionality, we can define the mapping function from concrete messages/ciphertexts to symbolic messages/ciphertexts. The ciphertexts generated by the adversary in local are considered as "invalid" and mapped to a special symbol called the "garbage" symbol.

In contrast, in this work based on the EUC framework, we cannot model cryptographic primitives as ideal functionalities because no EUC-secure cryptographic primitives such as public-key encryption and digital signature have been known. Thus, we need to develop a new method to map concrete messages to symbolic ones.

Our method is to observe all calls to the shared functionality, which models PKI. We use the keys generated by the shared functionality to check the validity of ciphertexts and signatures. If a ciphertext (resp., a signature) is decrypted (resp., verified) by a valid key, then it is mapped to a corresponding symbolic message, otherwise, the "garbage" symbol. In this way, even if the used cryptographic primitives are not EUC-secure, we success to establish a correspondence between two models in a proof of the mapping lemma, and guarantee the computational soundness of the symbolic model.

4 Conclusion

In this report, we presented essential parts of our contributions on protocol analysis in [4][5]. The first result we report here is the characterization of the common feature of NIMPC protocols. We derived a lower bound on the communication complexity, which can be used for security analysis. The second is a formalization of PKI-based cryptographic protocols to automatically analyze the security.

The conventional analysis methods assume the security of used cryptographic primitives. However, recent vulnerabilities such as the Logjam attack in May 2015 [2] and the SLOTH attack in Jan. 2016 [3] allow man-in-the-middle attackers to use "export-grade" weak cipher suits in TLS (Transport Layer Security). Thus, a possible future work is to develop analysis methods that include such attack methodologies.

References

1 Möller B., Duong T., and Kotowicz K., "This POODLE bites: SSL 3.0 fallback (security advisory)," https://www.openssl.org/~bodo/ssl-poodle.pdf, 2014.

2 Adrian D., Bhargavan K., Durumeric Z., Gaudry P., Green M., Halderman J.A., Heninger N., Springall D., Thomè E., Valenta L., VanderSloot B., Wustrow E., Zanella-Bèguelin S., and Zimmermann P., "Imperfect forward secrecy: How Diffie-Hellman fails in practice," In: The 22nd ACM Conference on Computer and Communications Security (ACM CCS 2015), pp.5–17, 2015.

3 Bhargavan K. and Leurent G., "Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH," In: The 23nd Annual Network and Distributed System Security Symposium 2016 (NDSS 2016).

4 Yoshida M. and Obana S., "On the (in) efficiency of non-interactive secure multiparty computation," In: The 19th Annual International Conference on Information Security and Cryptology (ICISC 2015), LNCS, vol.9558, pp.185–193, Springer, Heidelberg, 2016.

5 Yoshida M., Suzuki I., and Fujiwara T., "A symbolic model for an externalized universally composable framework," JSIAM Spring Conference 2014, Formal Approach to Information Security (FAIS), 2014.

6 Yao A.C., "Protocols for secure computations," In: The 23rd Annual Symposium on Foundations of Computer Science (FOCS '82), pp.160–164, 1982.

7 Chaum D., Crèpeau C., and Damgård I., "Multiparty unconditionally secure protocols," In: The 20th Annual ACM Symposium on Theory of Computing (STOC '88), pp.11–19, 1988.

8 Hirt M. and Maurer U., "Player simulation and general adversary structures in perfect multiparty computation," In: Journal of Cryptology, 13(1), pp.31–60, Springer, Heidelberg, 2000.

9 Beimel A., Ishai Y., Kushilevitz E., Meldgaard S., and Paskin-Cherniavsky A., "Non-interactive secure multiparty computation," In: The 34th Annual International Cryptology Conference (CRYPTO 2014), LNCS, vol.8617, pp.387–404, Springer, Heidelberg, 2014.

10 Canetti R. and Herzog J., "Universally composable symbolic analysis of mutual authentication and key-exchange protocols," In: The Third Theory of Cryptology Conference (TCC 2006), LNCS, vol.3876, pp.380–403, Springer, Heidelberg, 2006.

11 Canetti R., "Universally composable security: A new paradigm for cryptographic protocols," In: The 42nd Annual Symposium on Foundations of Computer Science (FOCS 2001), pp.136–145, IEEE Computer Society, 2001.

12 Dahl M. and Damgård I., "Universally composable symbolic analysis for two-party protocols based on homomorphic encryption," In: The 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt 2014), LNCS, vol.8441, pp.695–712, Springer, Heidelberg, 2014.

13 Canetti R. and Rabin T., "Universal composition with joint state," In: The 23rd Annual International Cryptology Conference (CRYPTO 2003), LNCS, vol.2729, pp.265–281, Springer, Heidelberg, 2003.

14 Canetti R., Dodis Y., Pass R., and Walfish S., "Universally composable security with global setup," In: The Fourth Theory of Cryptology Conference (TCC 2007), LNCS, vol.4392, pp.61–85, Springer, Heidelberg, 2007.

Maki YOSHIDA, Ph.D.

Senior Researcher, Security Fundamentals Laboratory, Cybersecurity Research Institute
Information Security, Cryptography, Information Hiding