Jana Dittmann
Stefan Katzenbeisser
Andreas Uhl (Eds.)

LNCS 3677

Communications and Multimedia Security

9th IFIP TC-6 TC-11International Conference, CMS 2005
Salzburg, Austria, September 2005
Proceedings

ifip

Springer

Jana Dittmann Stefan Katzenbeisser
Andreas Uhl (Eds.)

Communications and Multimedia Security

9th IFIP TC-6 TC-11International Conference, CMS 2005
Salzburg, Austria, September 19 – 21, 2005
Proceedings

Springer

Volume Editors

Jana Dittmann
Otto-von-Guericke-Universität Magdeburg
Institut für Technische und Betriebliche Informationssysteme
Universitätsplatz 1, 39106 Magdeburg, Germany
E-mail: Jana.Dittmann@iti.cs.uni-magdeburg.de

Stefan Katzenbeisser
Technische Universität München
Institut für Informatik
Boltzmannstrasse 3, 85748 Garching, Germany
E-mail: katzenbe@in.tum.de

Andreas Uhl
Universität Salzburg
Department of Scientific Computing
Jakob Haringer Strasse 2, A-5020 Salzburg, Austria
E-mail: uhl@cosy.sbg.ac.at

Library of Congress Control Number: 2005931928

CR Subject Classification (1998): C.2, E.3, D.4.6, H.5.1, K.4.1, K.6.5, H.4

ISSN 0302-9743
ISBN-10 3-540-28791-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-28791-9 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11552055 06/3142 5 4 3 2 1 0

Preface

It is our great pleasure to present the proceedings of the 9th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2005), which was held in Salzburg on September 19–21, 2005. Continuing the tradition of previous CMS conferences, we sought a balanced program containing presentations on various aspects of secure communication and multimedia systems. Special emphasis was laid on papers with direct practical relevance for the construction of secure communication systems.

The selection of the program was a challenging task. In total, we received 143 submissions, from which 28 were selected for presentation as full papers. In addition to these regular presentations, the CMS conference featured for the first time a "work in progress track" that enabled authors to report preliminary results and ongoing work. These papers were presented in the form of a poster session during the conference; an extended abstract of the posters appears in this proceedings volume. From all papers submitted to the CMS conference, the program committee chose 13 submissions for inclusion in the work in progress track.

In addition to regular presentations, CMS 2005 featured a special session on XML security, containing both contributed and invited talks. This special session was jointly organized by Rüdiger Grimm (TU Ilmenau, Germany) and Jörg Schwenk (Ruhr-Universität Bochum, Germany). Their assistance in organizing CMS 2005 was greatly appreciated.

Besides the above mentioned presentations, the scientific program of CMS 2005 featured three invited speakers: Christian Cachin (IBM Zürich), with a talk about the cryptographic theory of steganography, Ton Kalker (HP Labs), with a survey talk on recent trends in the field of Digital Rights Management, and Ingemar Cox (University College London), with a talk about robust watermarking schemes.

We want to thank all contributors to CMS 2005. In particular, we are grateful to the authors and invited speakers for contributing their latest work to this conference, as well as to the PC members and external reviewers for their critical reviews of all submissions. Finally, special thanks go to the organizing committee who handled all local organizational issues and provided us with a comfortable location and a terrific social program. For us, it was a distinct pleasure to serve as program chairs of CMS 2005.

We hope that you will enjoy reading these proceedings and that they will be a catalyst for your future research in the area of multimedia security.

July 2005

Jana Dittmann
Stefan Katzenbeisser
Andreas Uhl

9th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security

September 19–21, 2005, Salzburg (Austria)

Program Chairs

Jana Dittmann, Otto-von-Guericke Universität Magdeburg, Germany
Stefan Katzenbeisser, Technische Universität München, Germany
Andreas Uhl, Universität Salzburg, Austria

IFIP TC-6 TC-11 Chairs

Otto Spaniol, RWTH Aachen, Germany
Leon Strous, De Nederlandsche Bank, The Netherlands

Program Committee

André Adelsbach, Ruhr-Universität Bochum, Germany
Elisa Bertino, University of Milan, Italy
Carlo Blundo, UNISA, Italy
Christian Cachin, IBM Zürich, Switzerland
Ingemar J. Cox, University College London, UK
David Chadwick, University of Kent, UK
Bart de Decker, KU Leuven, Belgium
Yves Deswarte, LAAS, France
Elke Franz, TU Dresden, Germany
Miroslav Goljan, SUNY Binghamton, USA
Patrick Horster, Universität Klagenfurt, Austria
Ton Kalker, HP Labs, USA
Stephen Kent, BBN Technologies, USA
Klaus Keus, BSI, Germany
Herbert Leitold, A-SIT, Austria
Nasir Memon, Polytechnic University, USA
Sead Muftic, Stockholm University, Sweden
Fernando Perez-Gonzalez, University of Vigo, Spain
Günter Pernul, Universität Regensburg, Germany
Reinhard Posch, Technische Universität Graz, Austria
Bart Preneel, KU Leuven, Belgium
Claus Vielhauer, Otto-von-Guericke University Magdeburg, Germany
Moti Young, Columbia University, USA

Local Organization

Dominik Engel
Roland Norcen
Helma Schöndorfer
Michael Tautschnig
Andreas Uhl

External Reviewers

Carlos Aguilar-Melchor
Felix Balado
Lejla Batina
Yannick Chevalier
Stelvio Cimato
Pedro Comesana
Peter Danner
Paolo D'Arco
Liesje Demuynck
Claudia Diaz
Kurt Dietrich
Wolfgang Dobmeier
Anas Abou El Kalam
Martin Feldhofer
Jessica Fridrich
Alban Gabillon
Sebastian Gajek
Steven Galbraith
Clemente Galdi
Jörg Gilberg
Ulrich Greveler
Hazem Hamed
Mark Hogan
Yongdae Kim
Franz Kollmann
Klaus Kursawe
Mario Lamberger
Peter Lipp
Mark Manulis
Björn Muschall
Vincent Naessens
Vincent Nicomette
Rodolphe Ortalo
Elisabeth Oswald
Federica Paci
Udo Payer
Luis Perez-Freire
Thomas Popp
Torsten Priebe
Markus Rohe
Thomas Rössler
Heiko Rossnagel
Martin Schaffer
Peter Schartner
Christian Schlaeger
Stefaan Seys
Dieter Sommer
Anna Squicciarini
Hung-Min Sun
Yagiz Sutcu
Ingrid Verbauwhede
Frederik Vercauteren
Tine Verhanneman
Kristof Verslype
Ivan Visconti
Ron Watro
Johannes Wolkerstorfer
Peiter Zatko

Table of Contents

Applied Cryptography

DRM & E-Commerce

Media Encryption

Multimedia Security

Privacy

Biometrics & Access Control

Network Security

Mobile Security

Work in Progress Track

Special Session: XML Security

Fast Contract Signing with Batch Oblivious Transfer

Ľubica Staneková[1],* and Martin Stanek[2],**

[1] Department of Mathematics, Slovak University of Technology,
Radlinského 11, 813 68 Bratislava, Slovakia
ls@math.sk

[2] Department of Computer Science, Comenius University,
Mlynská dolina, 842 48 Bratislava, Slovakia
stanek@dcs.fmph.uniba.sk

Abstract. Oblivious transfer protocol is a basic building block of various cryptographic constructions. We propose a novel protocol – batch oblivious transfer. It allows efficient computation of multiple instances of oblivious transfer protocols. We apply this protocol to improve the fast simultaneous contract signing protocol, recently proposed in [11], which gains its speed from computation of time-consuming operations in advance. Using batch oblivious transfer, a better efficiency can be achieved.

1 Introduction

Oblivious transfer is a cryptographic protocol in which one party (usually called sender) transfers one of two strings to the other party (usually called chooser). The transfer should have the following properties: The chooser should obtain the string of his/her choice but not the other one, and the sender should be unable to identify the chooser's choice. Oblivious transfer is used as a key component in many cryptographic applications, such as electronic auctions [12], contract signing [4,11], and general multiparty secure computations [8]. Many of these and similar applications make intensive use of oblivious transfer. Therefore, efficient implementation of oblivious transfer can improve the overall speed and applicability of various protocols.

Batch variants of various cryptographic constructions are useful for decreasing computational costs. A batch variant of RSA, suitable for fast signature generation or decryption, was proposed by Fiat [5]. Batch verification techniques [1] can be used for efficient proofs of correct decryptions in threshold systems with applications to e-voting and e-auction schemes.

Simultaneous contract signing is a two-party cryptographic protocol, in which two mutually suspicious parties A and B wish to exchange signatures on a contract. Intuitively, a fair exchange of signatures is one that avoids a situation

* Supported by APVT 023302.

** Supported by VEGA 1/0131/03.

J. Dittmann, S. Katzenbeisser, and A. Uhl (Eds.): CMS 2005, LNCS 3677, pp. 1–10, 2005.

where A can obtain B's signature while B cannot obtain A's signature and vice-versa. There are two types of contract signing protocols: the ones that use trusted third party either on-line or off-line [6], and protocols without trusted third party [4,7]. Protocols without trusted third party are based on gradual and verifiable release of information. Hence, if one participant stops the protocol prematurely, both participants have roughly the same computational task in order to find the other participant's signature.

Recently, a contract signing protocol that allows pre-computation of significant part of the most time consuming operations in advance was proposed in [11]. The protocol makes an extensive use of oblivious transfers (its security depends on the security of oblivious transfers) in each protocol run.

Motivation. Oblivious transfer is frequently used in cryptographic protocols. There are many protocols in which a large number of oblivious transfers is employed in a single protocol instance. Therefore, an efficient implementation of oblivious transfer is a natural way to improve the efficiency of such protocols.

Our Contribution. We present a batch RSA oblivious transfer protocol where multiple independent instances of oblivious transfers can be computed efficiently. The security of the protocol is based on RSA assumption, and we prove it in the random oracle model.

We compare actual implementation of batch RSA oblivious transfer protocol with standard RSA oblivious transfer [11], and oblivious transfer based on the computational Diffie-Hellman assumption [13].

We show the usefulness and applicability of our proposal and improve the simultaneous contract signing protocol [11]. The use of batch RSA oblivious transfers instead of pre-computed oblivious transfers leads to more efficient protocol. Both settings were implemented and compared to illustrate exact decrease of computational costs.

Related Work. The efficiency of computing oblivious transfer influences the overall efficiency of many protocols. Our batch RSA oblivious transfer is a modification of the RSA oblivious transfer protocol from [11]. Other constructions of oblivious transfer employ some kind of ElGamal encryption or computational Diffie-Hellman assumption [13].

Similar problem of amortizing the cost of multiple oblivious transfers, based on computational Diffie-Hellman assumption, has been considered by Naor and Pinkas [13]. We compare our approach with their constructions in Sect. 4.

Our security proofs for batch RSA oblivious transfers make use of random oracles. The application of random oracles in the security analysis of cryptographic protocols was introduced by Bellare and Rogaway [2]. Security proofs in a random oracle model substitute a hash function with ideal, truly random function. This approach has been applied to many practical systems, where the ideal function must be instantiated (usually as a cryptographically strong hash function). Recently, an interesting discussion about plausibility of security proofs in the random oracle model appeared in [10].

The paper is structured as follows. Section 2 presents our main result, the batch RSA oblivious transfer, and its implementation. The protocol for contract signing is described in Sect. 3. We analyse an actual implementation of batch RSA oblivious transfer and the savings of computational costs resulting from its application in Sect. 4.

2 Batch Oblivious Transfer

Oblivious Transfer (OT) protocol, more specifically OT_1^2 protocol, allows two parties (sender and chooser) to solve the following problem. The sender has two strings m_0 and m_1 and transfers one of them to the chooser in accordance with the following conditions:

- the chooser selects a particular m_b which he wishes to obtain ($b \in \{0, 1\}$);
- the chooser does learn nothing about m_{1-b};
- the sender does not know which m_b was transferred.

We modify and extend construction of RSA-based OT_1^2 protocol from [11]. Most oblivious transfer protocols employ some kind of ElGamal encryption. This results in increased computational overhead as the chooser must perform at least one modular exponentiation. Using RSA-based oblivious transfer allows to reduce the chooser's complexity, since the public exponent can be made small. Moreover, RSA decryption with distinct private exponents can be implemented efficiently, leading to Batch RSA [5]. We use this idea for further improvement of computational complexity of RSA-based oblivious transfer.

We employ the following notation through the rest of the section. Let $n = p \cdot q$ be an RSA public modulus (i.e. a product of two distinct primes p and q) and let e, d denote public and private exponents, respectively. Let $Z_n = \{0, 1, \ldots, n-1\}$ and let Z_n^* be the set of all numbers from Z_n relatively prime to n. All computations in protocol descriptions are defined over Z_n, the only exception is bitwise xor operation $\oplus$. We will omit stating explicitly that our operations in the paper are mod n whenever it is clear from the context. The hash function H is modelled as a truly random function (random oracle, see [2]) in the security analysis. For simplicity we write $H(a_1, \ldots, a_l)$ for the hash function applied to the concatenation of l-tuple $(a_1, \ldots, a_l)$. Random, uniform selection of x from the set A is denoted by $x \in_R A$.

We assume the sender (S in protocol description) generates the instance of RSA system and the chooser (C) already has a valid public key of the sender (i.e. a pair (n, e)). Moreover, we assume that the length of H output is not shorter than strings m_0 and m_1. Recall, $b \in \{0, 1\}$ denotes the index of string, which the chooser wants to obtain.

2.1 RSA Oblivious Transfer

The RSA oblivious transfer protocol [11] is a modification of the protocol [9]. Since the protocol is executed multiple times a sufficiently long random string R (chosen by sender) is used to distinguish the instances of the protocol.

1. S → C : $C \in_R Z_n^*$
2. C → S : $x' = x^e C^b$, where $x \in_R Z_n$.
3. S → C : R, E_0, E_1,
   where ciphertexts E_0, E_1 of strings m_0, m_1 are computed as follows:
   $$E_0 = H(R, x'^d, 0) \oplus m_0; \qquad E_1 = H(R, (x'C^{-1})^d, 1) \oplus m_1.$$
4. The chooser decrypts m_b from E_b: $m_b = E_b \oplus H(R, x, b)$.

Since the value x' is uniformly distributed in Z_n, the chooser's security is protected in an information-theoretic sense – the sender cannot determine b, even with infinite computational power. The sender's security can be proved in the random oracle model under RSA assumption. The protocol allows precomputation of value $(C^{-1})^d$, thus allowing efficient implementation of protocols, where multiple instances of oblivious transfer are required.

Remark 1. Roughly the same efficiency can be obtained (without any precomputation) by generating C^d randomly first and computing C by exponentiation to the short public exponent. This possibility was neglected by the authors of this protocol. Batch oblivious transfer is even more efficient, as we will see later.

2.2 Batch RSA Oblivious Transfer

The main observation regarding efficiency of RSA oblivious transfer is the fact that multiple parallel executions can use distinct private exponents. This allows to reduce computational complexity of sender using techniques of Batch RSA.

We assume that L oblivious transfers should be performed. Let $m_{i,0}$, $m_{i,1}$ (for $0 \leq i < L$) be input strings for i-th oblivious transfer. Similarly, $b_0, \ldots, b_{L-1}$ are indices of those strings, which the chooser wants to obtain. The sender selects L distinct small public RSA exponents $e_0, \ldots, e_{L-1}$, each one relatively prime to $(p-1)(q-1)$, and computes corresponding private exponents $d_0, \ldots, d_{L-1}$. For efficient implementation the public exponents must be relatively prime to each other and $e_i = O(\log n)$, for $i = 0, \ldots, L-1$.

The protocol executes L separate instances of oblivious transfer:

1. S → C : $C_0, C_1, \ldots, C_{L-1} \in_R Z_n^*$
2. C → S : $x'_0, x'_1, \ldots, x'_{L-1}$,
   where $x'_i = x_i^{e_i} C_i^{b_i}$ and $x_i \in_R Z_n$, for $i = 0, \ldots, L-1$.
3. S → C : $\{R_i, E_{i,0}, E_{i,1}\}_{0 \leq i < L}$,
   where ciphertexts $E_{i,0}$, $E_{i,1}$ of strings $m_{i,0}$, $m_{i,1}$ are computed as follows:
   $$E_{i,0} = H(R_i, (x'_i)^{d_i}, i, 0) \oplus m_{i,0};$$
   $$E_{i,1} = H(R_i, (x'_i C_i^{-1})^{d_i}, i, 1) \oplus m_{i,1}.$$
4. The chooser decrypts $m_{i,b_0}, \ldots, m_{i,b_{L-1}}$ from $E_{i,b_0}, \ldots, E_{i,b_{L-1}}$:
   $$m_{i,b_i} = E_{i,b_i} \oplus H(R_i, x_i, i, b_i), \quad \text{for } i = 0, \ldots, L-1.$$

One can easily check the correctness of the decryption:

$$\begin{aligned} E_{i,b_i} \oplus H(R_i, x_i, i, b_i) &= H(R_i, (x'_i C_i^{-b_i})^{d_i}, i, b_i) \oplus m_{i,b_i} \oplus H(R_i, x_i, i, b_i) \\ &= H(R_i, (x_i^{e_i} C_i^{b_i} C_i^{-b_i})^{d_i}, i, b_i) \oplus m_{i,b_i} \oplus H(R_i, x_i, i, b_i) \\ &= m_{i,b_i} \end{aligned}$$

Security. The chooser's objective is to hide values $b_0, \ldots, b_{L-1}$ from the sender. The values x'_i are uniformly distributed in Z_n. Thus, the sender cannot compute b_i, even with unrestricted computational power – for each transmitted L-tuple $x'_0, \ldots, x'_{L-1}$ and every possible selection of values $b_0, \ldots, b_{L-1}$ there exist suitable choices $x_0, \ldots, x_{L-1} \in Z_n$ (easily computed by the sender himself):

$$x_0 = (x'_i \cdot C_i^{-b_i})^{d_i}, \qquad \ldots, \qquad x_{L-1} = (x'_{L-1} \cdot C_{L-1}^{-b_{L-1}})^{d_{L-1}}.$$

Hence, all combinations of values $b_0, \ldots, b_{L-1}$ are equiprobable and the sender cannot identify the correct one. The chooser's security is protected unconditionally.

The sender's objective is to hide one string from every pair $m_{i,0}$, $m_{i,1}$ (not knowing which one exactly). We prove this security property of the protocol in random oracle model, where the hash function H is modelled as a random function.

We compare the protocol with the ideal implementation (model). The ideal model uses a trusted third party that receives all $m_{i,0}$ and $m_{i,1}$ from the sender and $b_0, \ldots, b_{L-1}$ from the chooser. After obtaining all inputs, the trusted third party sends the chooser m_{i,b_i}, for $0 \leq i < L$. The ideal model hides the values $m_{i,1-b_i}$ perfectly – no adversary substituting the chooser can learn anything about hidden values. The actual protocol should be comparable with the ideal model in the following sense (for extensive study of various definitions of protocol security in the ideal model see [3]):

> For every distribution on the inputs $\{m_{i,0}, m_{i,1}\}_{0 \leq i < L}$ and any probabilistic polynomial adversary A substituting the chooser in the actual protocol there exists a probabilistic polynomial simulator S_A in the ideal model such that outputs of A and S_A are computationally indistinguishable.

Since the ideal model is secure and outputs of A and S_A are indistinguishable, one can conclude that A does not learn more than allowed by security requirements.

The simulator S_A simulates both the sender and adversary A. Therefore, the verb "send" refers to writing data to input or reading data from output of simulated adversary.

1. S_A selects random $C_0, C_1, \ldots, C_{L-1} \in_R Z_n^*$ and sends them to A. It starts to simulate A on this input.

2. A sends values $x'_0, x'_1, \ldots, x'_{L-1} \in Z_n$ to S_A. These values can be computed by adversary A in any way (adversary does not need to follow the protocol).
3. S_A selects random strings $\{R_i, E_{i,0}, E_{i,1}\}_{0 \leq i < L}$ as "sender's answer" and sends them in response.
4. S_A continues the simulation of A and monitors all its queries to H. All queries have the form of a quadruple (R, x, i, b). We say that the quadruple (R, x, i, b) is valid if $R_i = R$ and $x'_i C_i^{-b} = x^{e_i}$. All queries not containing a valid quadruple are answered at random. If A asks for $H(R, x, i, b)$, where the argument is a valid quadruple, then S_A asks a trusted third party in the ideal model for $m_{i,b}$. The simulator sets $H(R, x, i, b) = E_{i,b} \oplus m_{i,b}$ to allow A to decrypt $E_{i,b}$ correctly. Whatever A outputs, so does S_A.

The distribution of simulated communication with the adversary A is identical to the distribution of real communication between the sender and A. The only exception is the case when A asks for any valid pair of quadruples $H(R, x, i, 0)$ and $H(R, x^*, i, 1)$, for $i \in \{0, \ldots, L-1\}$. In this case, the validity of the quadruples implies $x'_i = x^{e_i}$ and $x'_i C_i^{-1} = (x^*)^{e_i}$. It easily follows that $x \cdot (x^*)^{-1}$ is the decryption of C_i:

$$(x \cdot (x^*)^{-1})^{e_i} = x^{e_i} \cdot (x^*)^{-e_i} = x'_i \cdot (x'_i)^{-1} C_i = C_i.$$

The values C_i are chosen randomly by the simulator S_A. Hence, the adversary cannot construct a pair of valid quadruples, assuming the RSA assumption holds. Therefore the output of S_A cannot be distinguished from the output of A in the real communication with the sender.

Remark 2. Random strings R_i are used in the protocol to ensure distinct inputs of H in different invocations of the protocol.

Remark 3. Less direct construction would use triples $(R_i, (x'_i C_i^{-b_i})^{d_i}, b_i)$ instead of quadruples $(R_i, (x'_i C_i^{-b_i})^{d_i}, i, b_i)$. The simulator would determine the correct value of index i by testing validity of all potential triples.

Implementation. The most time-consuming part of the protocol is step 3, where the sender computes $2L$ RSA decryptions. The use of distinct pairs of encryption/decryption exponents enables to apply batch RSA decryption [5]. The sender needs to compute following decryptions in step 3:

$$(x'_i)^{d_i}, (x'_i C_i^{-1})^{d_i}, \qquad \text{for } i = 0, \ldots, L-1.$$

Certainly, only one decryption has to be computed for every i, namely $(x'_i)^{d_i}$. This follows from an observation that $(x'_i C_i^{-1})^{d_i} = (x'_i)^{d_i} (C_i^{d_i})^{-1}$, and C_i can be generated from randomly chosen $C_i^{d_i}$ by encrypting it: $(C_i^{d_i})^{e_i}$ (thus having decryption "for free"). Assuming small size of public (encryption) exponents, the computation can be implemented in such a way that L decryptions $(x'_i)^{d_i}$ require time asymptotically proportional to one decryption, see [5]. Notice, that small public exponents yield efficient implementation of the chooser's part of the protocol as well.