Fast Mutual Authentication Using RSA Moduli

Anatoly V. Anisimov, *Member, IEEE*, and Andrey Novokshonov

Abstract—We describe a fast three-round mutual authentication protocol for parties A and B belonging to the same coalition group. Parties A and B keep their own independent long-term private keys that are used in the process of authentication and can be used for other purposes. The scheme assumes an initial setup with a trusted third party T . This party initiates another secret information that includes factors of a large RSA modulus. For authentication, both parties must demonstrate each other the knowledge of their private keys without revealing them and the ability to factorize a large RSA modulus. Thus, the protocol based on the suggested scheme provides reciprocal authentication. The scheme possesses all desirable properties of an interactive proof, i.e., completeness, soundness, and zero-knowledge. The security of the protocol relies on assumptions of difficulty of the RSA factorization and existence of a cryptographic hash function.

Index Terms—authentication, Fiat-Shamir heuristic, interactive proof, RSA modulus, zero-knowledge

I. INTRODUCTION

MODERN technological systems fulfilling critically dependent tasks are marked by a rich variety of interaction types and exponentially growing complexity. Examples of such systems are various peer-to-peer networks, interacting human-machine and machine-to-machine complexes, military and corporative coalition units, financial systems including cryptocurrencies, smart cards, Internet of Things (IoT), and many others. Along with this, one can see the increasing activity of technologically elaborated adversarial attacks with the purpose to maliciously intrude into the work of such systems. In this regard, the speed and accuracy of authentication procedures are of special importance. In many cases, establishing the well-founded identity of an object demands essential computational and managerial efforts. In many critical cases, establishing secure authentication may be inconsistent with the necessity to immediately respond to an authentication query. Especially, this problem is important due to the ubiquitous advent of IoT with hundreds and thousands of interacting “computationally weak” devices. Thus, in many critical cases, a quite complex problem arises. On the one hand, it is necessary to obtain a reasonable, secure, and mathematically grounded object identification. On the other hand, it should be done in minimal admissible time.

Mutual authentication means the case when entities simultaneously identify each other executing the same authenticating protocol. Mutual authentication originates from

the network authentication client-server protocols first developed in the Needham-Schroeder protocol [1] that later evolved to the well-known system Kerberos. Mutual authentication is one of the most important parts of customized IoT systems that are marked by the vast variety of specific cases reflecting broad areas of applications. A comprehensive survey of this issue is given in [2], [3].

With the development of zero-knowledge interactive proofs, the authentication problem obtains a new lightening. In challenge-response zero-knowledge protocols, authentication is carried out through a verification process where a claimant, called a prover, must corroborate to a verifier the knowledge of some secret identifying information without revealing any part of it. Zero-knowledge property is a very strong security requirement. In many cases, such protocols are mathematically substantiated, but they are unilateral and cannot be automatically adapted to the mutual case [4].

The main feature of secure coalition authentication is the necessity for both parties not only to prove the knowledge of some secret identifying information but also to demonstrate the belonging to the same authorized group. This assumes the implicit or explicit impact of the trusted third party and demands a reciprocal zero-knowledge verification.

General interactive and non-interactive zero-knowledge protocols mainly concentrate on optimization proofs for general NP relations [5], [6], [7], [8]. Despite many impressive achievements in this direction, they are still quite complex in implementation and are not well suited for the immediate two-way authentication response.

Faster one-factor (only one secret value of the prover) authentication zero-knowledge protocols like the family of the Fiat-Shamir [9], [10], [11] and the Schnorr protocols [12], when applied to the coalition case, must maintain some extra interactions and setup computations. The security of the Fiat-Shamir protocol relies on the strong RSA assumption, i.e., the difficulty of factoring the product of two large prime numbers. The security of the Schnorr protocol is backed by the complexity of the discrete logarithm problem in a group. Using the Fiat-Shamir heuristic [9], these schemes can be converted into unilateral non-interactive protocols based on digital signatures that confirm the prover’s identity. For the mutual coalition authentication, this is not possible without the use of additional interacting and/or resetting procedures.

Manuscript received

A. V. Anisimov is with the Faculty of Computer Science and Cybernetics, Taras Shevchenko National University of Kyiv (e-mail: anatoly.v.anisimov@gmail.com).

A. Novokshonov is with the Department of Intellectualization of Information Technologies, International Research and Training Center for Information Technologies and Systems under NAS and MES of Ukraine (e-mail: andrey.novokshonov@ukr.net).

II. GENERAL DESCRIPTION OF THE $\alpha\beta$ -PROTOCOL

In this communication we propose a general solution for the fast and secure authentication problem for entities belonging to the same coalition group, i.e., we consider authentication proofs of the type “friend or foe”. Like the Fiat-Shamir protocol, this condition is equivalent to the natural assumption that both the prover A (Alice) and the verifier B (Bob) had access to the same mutually trusted center T that independently issued for them additional session keys and generated some new secret information. In our case, these are factors of a randomly chosen large RSA modulus. These factors can be found both by A and B only through a three-round interaction using their own private keys and combined keys generated by T . A and B must prove each other the knowledge of their private keys without revealing them, and, additionally, find factors of a randomly chosen large RSA modulus selected by the authorized coalition controller T . Also, we emphasize that for parties there is no need to publicly reveal the factors of the given RSA number. Symmetrically, parties A and B can exchange roles “prover-verifier”. Thus, the proposed scheme provides mutual authentication. Also, it possesses all desirable properties of interactive proofs such as completeness, soundness, and mutual zero-knowledge.

More detailed, during the setup stage, parties A and B independently present to the trusted center T their randomly chosen private keys s_1 and s_2 , respectively. The center T randomly generates a numerated list of RSA moduli $n = pq$ where p and q are prime numbers being coprime with s_1 and s_2 . The list of these moduli could be public or at least should be known to B . Also, the center T creates and distributes between A and B additional corresponding secret keys depending on multipliers of n , private keys s_1 , s_2 , and a random parameter a . Only through interactions using these combined keys and their own private keys, A and B can factorize the given RSA modulus. This way they demonstrate the knowledge of corresponding private keys and the ability to factorize the number n . The parameter a is a probabilistic seed for the verifier which starts the interaction. After the initial setup, the trusted center does not need to keep these secrets as well as prime factors p and q .

We name the proposed scheme the $\alpha\beta$ -protocol because it uses two basic simple interconnected procedures called α and β , $\alpha(x, y, z) = x^{-1}y^{-1} \bmod z$, $\beta(x, u, v, z) = xuv \bmod z$. For the procedure α , the necessary condition for arguments is $\gcd(xy, z) = 1$. We note that if in $\beta(x, u, v, z)$ $u = yw$, $v = \alpha(x, y, z) = x^{-1}y^{-1} \bmod z$ and integers x, y, w are less than z then $\beta(x, u, v, z)$ returns w which is the factor of u . The procedure α fulfills modular inversions and it is used by the center T . The procedure β executes only one modular multiplication of three numbers. It is used by A and B for integer factorization.

We describe two variants of the $\alpha\beta$ -protocol: a basic protocol and its extension. The basic protocol is unilateral. In the basic variant, A proves to B that she knows her private value s_1 and can factorize a randomly chosen large number n without revealing s_1 and factors of n . The number n is a mutual input

for A and B .

After the initial setup, the basic protocol uses only two interactions and fixed roles for A and B : A is a prover, and B is a verifier. Augmented with the third interaction, which is the additional response of B , the modified protocol provides a reciprocal authentication of both parties.

The security of the protocol is based on the assumptions of difficulty of RSA factorization and the existence of a cryptographic hash function. Also, in practice, both parties can easily calculate the expected response time of a counterparty and reject in the case of exceeding this time.

Computationally, the scheme is very simple. In the restricted basic variant, the verifier B authenticates the prover A only using two rounds. In an extended reciprocal variant, they both verify each other using the minimal number of three communication rounds: 1) commitment/challenge, 2) response/commitment/challenge, 3) response. During rounds, interacting parties execute only one modular multiplication of three integers and a hashing procedure.

We consider the $\alpha\beta$ -protocol as a three-factor authentication scheme because it uses three independently created sets of unique factors: two private keys of A and B and a set of combined keys for integer factorization randomly generated by the trusted third party.

The proposed version of the $\alpha\beta$ -protocol originates from considerations presented in [13]. Comparing with [13], it uses a very different approach based on the interactive proof techniques.

III. THE BASIC $\alpha\beta$ -PROTOCOL

By default, we assume that all numbers considered hereinafter are positive natural integers.

Let 1^t be a security parameter, h a public hash function acting on strings, $h: \{0,1\}^* \rightarrow \{0,1\}^t$. For integers x, y, z when writing $h(x, y, z)$, we assume that arguments x, y , and z , given in binary, are concatenated. We note that if x, y, z are integers and bitlengths of z and y are given then from the concatenated string $x * y * z$ the number x could be restored unambiguously.

While establishing properties of the developed $\alpha\beta$ -protocol, we use main properties of hash functions: 1) it is infeasible to compute the given image $h(z)$ without getting z ; 2) having z , it is infeasible to find another $\tilde{z}$ such that $h(z) = h(\tilde{z})$; 3) $h(z)$ can be considered as statistically independent of z and uniformly distributed. The last property is used in the cryptographic random oracle model.

The basic $\alpha\beta$ -protocol consists of the initial setup interaction with the mutually trusted third party T and two interaction rounds between the prover A and the verifier B .

0. Initial setup.

01. T randomly chooses two prime numbers p and q of the bitlength t and constructs the RSA product $n = pq$.

02. A and B independently select random long-term secret keys s_1 and s_2 , respectively, of the bitlength $t + 1$. A and B send s_1 and s_2 to T .

$A \rightarrow T: s_1, B \rightarrow T: s_2$.

03. T randomly selects an integer a such that $\gcd(a, s_1) =$

1. By agreement with T , this number may be chosen by B .

Along with generating n and a , the center T computes associated values $b = \alpha(a, q, s_1) = a^{-1}q^{-1} \bmod s_1$, $c = h(b, p, s_1)$, and $d = c^{-1}p^{-1} \bmod s_2$, $\gcd(c, s_2) = 1$.

04. T sends the number b to A and the pair (a, d) to B .

$T \rightarrow B: b, T \rightarrow A: a, d$.

T may create several such numbers n . For each n , several associated values a, b, c, d may exist. It is convenient to consider these values to be indexed by corresponding arrays.

Parties A and B keep their numbers b and (a, d) , respectively, as additional secrets. We assume that during the setup stage all parties A, B , and T behave honestly.

1. B starts the interaction with A . From the list of given RSA moduli, B randomly chooses a modulus $n = pq$ and starts the interaction pointing out which n he chooses and sending to A the corresponding value a associated with n .

$B \rightarrow A: a$.

2. (Weak acceptance of B). Upon obtaining a from B , A computes $\beta(a, b, n, s_1) = abn \bmod s_1 = p$. This number should be a divisor of n . If so, A weakly accepts B and continues. Otherwise, A rejects.

3. A computes $c = h(b, p, s_1)$ and sends it to B .

$A \rightarrow B: c$.

4. (Acceptance of A). Upon receiving c , B calculates $\beta(c, d, n, s_2) = cdn \bmod s_2$ and accepts if and only if β returns a divisor of n . Otherwise, B rejects the proof.

The protocol is successful if A weakly accepts B and B accepts A .

It is convenient to assume that, in the case of rejecting, a party sends a special message $\emptyset$ to its counterpart.

IV. PROPERTIES OF THE BASIC $\alpha\beta$ -PROTOCOL

Theorem 1. The basic $\alpha\beta$ -protocol has the following properties: completeness, soundness, and honest-verifier zero-knowledge.

Proof. We remind that in the basic $\alpha\beta$ -protocol A is the prover and B is the verifier.

A. Completeness

If A and B are honest parties and follow the protocol, then B always accepts the proof with probability 1. Indeed, from the structure of b , $b = a^{-1}q^{-1} \bmod s_1$, it follows that $abn \bmod s_1 = p$. Analogous statement holds true for c, d, n, s_2 and q , $cdn \bmod s_2 = q$ if and only if $d = c^{-1}p^{-1} \bmod s_2$. If A and B are honest and follow the protocol then through interactions both A and B can easily factorize n using their private moduli s_1 and s_2 , respectively. A computes p (step 2) using given b, n , and the incoming number a and B computes q (step 3) using given d, n , and the incoming c . Also, A using the given number b and the derived p can compute $c = h(b, p, s_1)$. Thus, if A and B follow the protocol then B always accepts the authentication proof of A , and A weakly accepts B .

At step 2 of the protocol, B demonstrates to A only the knowledge of the right integer a that matches b and n but neither the knowledge of his private secret s_2 nor the ability to factorize n . We consider this as a weak identity proof of B .

B. Soundness

Soundness means that any dishonest prover A^* (a PPT Turing machine) impersonating A will be unable to convince the honest verifier B that she knows the secret s_1 belonging to A and factors of n in the case of having no information about these values except with negligible probability.

Assume that with non-negligible probability B accepts A^* . After obtaining n and a from B , A^* must respond sending c' to B . The honest verifier B can accept if and only if $c'dn \bmod s_2 = q$. Numbers d and n have been given to B by the trusted (honest) center T . This implies that they are true numbers created during the setup stage. It follows that $c'dn \bmod s_2 \equiv c'(c^{-1}p^{-1})pq \bmod s_2 = q$. This implies $c'c^{-1}p^{-1}p \bmod s_2 \equiv 1 \bmod s_2$ and we get $c'c^{-1} \equiv 1 \bmod s_2$. This implies that $c' \equiv c \bmod s_2$. Since the bitlength of c' should be less than that of s_2 , we conclude that $c' = c = h(b, p, s_1)$. Properties of a hash function imply that if A^* can output $c = h(b, p, s_1)$ with non-negligible probability, then during its activity A^* should get b, p , and s_1 with non-negligible probability. This allows constructing a knowledge extractor simulating machine.

For extracting knowledge (b, p, s_1) , we construct a simulating PPT machine M fed by the program of A^* . At the first round, M simply repeats actions of A^* and simultaneously records on the tape its internal coin tosses until obtaining the output value c' that should be sent to B . Only the case $c' = c = h(b, p, s_1)$ allows B to get the factor of n equal to q and come to the accepting state. The function h is a one-way hash function. This implies that A^* can calculate c only when she got (b, p, s_1) at one of the preceding steps. Instead of sending c to B produced by A^* , machine M makes a copy of c and at the next round will use it for the search of (b, p, s_1) .

At the second round, M rewinds to the starting state of A^* and using the record of its coin tosses repeats the behavior of A^* with the same random coins but additionally makes the search for b, p, s_1 . During this stage of execution, upon obtaining any value v , M checks if $h(v) = c$. After finding such v , M uses it for extracting s_1 (the last $t + 1$ bits of v) and finding p (the last t remaining prefix bits in the string v after extracting s_1).

Thus, this way with non-negligible probability M extracts secret knowledge of A .

C. Honest-verifier zero-knowledge

Zero-knowledge means that there exists a PPT simulating Turing machine Sim that without knowledge of the value s_1 using the program of the verifier B and information produced by the trusted center T that is statistically independent of s_1 generates the transcript of communications between A and B indistinguishable from the original interactions between A and B in the protocol. So, Sim knows (keeps on its input tape): from $T - p, q$, and n , from $B - a, s_2$, and d . In our case, both A and B are honest and each singular interaction transcript has the form (a, c) , where $c = h(b, p, s_1)$. So, Sim takes $d = c^{-1}p^{-1} \bmod s_2$ extracted from B and computes $c' = d^{-1}p^{-1} \bmod s_2$. For any x the length of $h(x)$ is t , and the length

of c' is less than that of s_2 . It follows that $c' \equiv c = h(b, p, s_1)$. This implies that $c' = c$.

Output generated by Sim on the input (n, a) is the pair (a, c) which coincides with the transcript of the interaction of A and B . All steps performed by Sim do not use any information connected with s_1 and are time-bounded by the fixed polynomial of t . The machine B is also assumed to be a PPT Turing machine then so is the machine Sim .

This ends the proof of Theorem 1.

V. MUTUAL $\alpha\beta$ -PROTOCOL

This is an extension of the basic $\alpha\beta$ -protocol. At the initial setup stage, in addition to integers a, b, d generated by the trusted center T , the protocol performs as follows.

05. T computes integers $e = h(d, q, s_2)$ and $g = \alpha(e, q, s_1) = e^{-1}q^{-1} \bmod s_1$. The number e should be coprime with s_1 .

06. The trusted center T sends g to A .

$T \rightarrow A: g$.

To obtain the reciprocal property, we add steps 4 and 5 to the basic $\alpha\beta$ -protocol.

4. B calculates $e = h(d, q, s_2)$ and sends e to A .

$B \rightarrow A: e$.

5. A calculates $p' = \beta(e, g, n, s_1) = egn \bmod s_1$. If p' is equal to p then A accepts B .

The protocol is successful if B accepts A (step 3) and A accepts B (step 5). The mutual $\alpha\beta$ -protocol is shown in Fig. 1.

Corollary 1. The mutual $\alpha\beta$ -protocol is complete, sound, and zero-knowledge.

Proof. In the extended variant, we can consider the $\alpha\beta$ -protocol as a proof not only authenticating A by B but also as a proof authenticating B by A . Indeed, we note that in the augmented variant, the scheme is symmetrical with respect to the roles "prover-verifier". Starting from round 2 (sending c to B) and obtaining the response from B at round 4, party A plays the role of the verifier, and B fulfilling steps 3 and 4 acts as the prover. Reciprocal soundness means that both A and B behave honestly. Considering Theorem 1 in the extended variant, the authentication of A and B is reciprocal, complete, sound, and zero-knowledge.

Fig. 1. Mutual $\alpha\beta$ -protocol

REFERENCES

- [1] R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers," *Commun. ACM*, vol. 21, no. 12, pp. 993–999, Dec. 1978, 10.1145/359657.359659.
- [2] N. Li, D. Liu, and S. Nepal, "Lightweight Mutual Authentication for IoT and Its Applications," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 4, pp. 359–370, Dec. 2017, 10.1109/TSUSC.2017.2716953.
- [3] T. Yang, G. Zhang, L. Liu, Y. Yang, S. Zhao, H. Sun, and W. Wang, "New Features of Authentication Scheme for the IoT: A Survey," in *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things (IoT S&P'19)*, New York, NY, USA, 2019, pp. 44–49.
- [4] A. Menezes, S. Vanstone, and P. Van Oorschot, "Identification and Entity Authentication" in *Handbook of Applied Cryptography*, 1st ed. USA: CRC Press, Inc., 1996, pp. 385–424.
- [5] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proceedings of the seventeenth annual ACM symposium on Theory of computing (STOC '85)*, New York, NY, USA, 1985, pp. 291–304.
- [6] M. Blum, B. Feldman, and T. Micali, "Non-interactive zero-knowledge and its applications," in *Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC '88)*, New York, NY, USA, 1988, pp. 103–112.
- [7] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems," *J. ACM*, vol. 38, no. 3, pp. 690–728, Jul. 1991, 10.1145/116825.116852.
- [8] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *J. Cryptol.*, vol. 7, no. 1, pp. 1–32, Dec. 1994, 10.1007/BF00195207.
- [9] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in *Proceedings on Advances in cryptography (CRYPTO '86)*, Santa Barbara, CA, USA, 1987, pp. 186–194.
- [10] U. Feige, A. Fiat, and A. Shamir, "Zero knowledge proofs of identity," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing (STOC '87)*, New York, NY, USA, 1987, pp. 210–217.
- [11] L. Guillou and J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory," in *Lecture Notes in Computer Science on Advances in Cryptology (EUROCRYPT '88)*, Davos, Switzerland, 1988, pp. 123–128.
- [12] C. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, Jan. 1991, 10.1007/BF00196725.
- [13] A. Anisimov, "Coalition Public-Key Cryptosystems," *Cybernetics and Systems Analysis*, vol. 37, no. 1, pp. 1–12, Jan. 2001, 10.1023/A:1016610214445.