Delegating a Product of Group Exponentiations with Application to Signature Schemes

Presenter: Giovanni Di Crescenzo

Perspecta Labs Inc.
(also doing business as Applied Communication Sciences)
(previously done business as Vencore Labs, Telcordia Applied Research, BellCoRe, Bell Labs)
E-Mail: *gdicrescenzo@perspectalabs.com*

Authors:
Giovanni Di Crescenzo, Perspecta Labs Inc. Basking Ridge, NJ, USA.
Matluba Khodjaeva, CUNY John Jay College of Criminal Justice. New York, NY, USA.
Delaram Kahrobaei, University of York. Heslington, York, UK.
Vladimir Shpilrain, City University of New York. New York, NY, USA.

06/26/2019

Introduction and Motivation

- **Group exponentiation** is a cornerstone operation in many public-key cryptographic protocols (e.g. RSA, DHKE, DSA, etc.)

Introduction and Motivation

- **Group exponentiation** is a cornerstone operation in many public-key cryptographic protocols (e.g. RSA, DHKE, DSA, etc.)
- To expand the applicability of these solutions to computationally weaker devices (e.g., passive RFID tags), it was advocated that most expensive operations like group exponentiation are delegated
  - from a computationally **weaker client** (i.e., a wireless, RFID device)
  - to a computationally **stronger server** (i.e., a cloud server).

Introduction and Motivation

- **Group exponentiation** is a cornerstone operation in many public-key cryptographic protocols (e.g. RSA, DHKE, DSA, etc.)
- To expand the applicability of these solutions to computationally weaker devices (e.g., passive RFID tags), it was advocated that most expensive operations like group exponentiation are delegated
  - from a computationally **weaker client** (i.e., a wireless, RFID device)
  - to a computationally **stronger server** (i.e., a cloud server).
- Preliminary solutions to this problem considered mostly honest servers or multiple servers of which at least one is honest. In the case of a **single**, possibly **malicious server**, this problem has remained open since a formal cryptographic model was introduced [HL'05].
- In [DKKS'17] we solved this problem for a large class of cyclic groups.

Introduction and Motivation

- **Group exponentiation** is a cornerstone operation in many public-key cryptographic protocols (e.g. RSA, DHKE, DSA, etc.)
- To expand the applicability of these solutions to computationally weaker devices (e.g., passive RFID tags), it was advocated that most expensive operations like group exponentiation are delegated
  - from a computationally **weaker client** (i.e., a wireless, RFID device)
  - to a computationally **stronger server** (i.e., a cloud server).
- Preliminary solutions to this problem considered mostly honest servers or multiple servers of which at least one is honest. In the case of a **single**, possibly **malicious server**, this problem has remained open since a formal cryptographic model was introduced [HL'05].
- In [DKKS'17] we solved this problem for a large class of cyclic groups.
- In this paper, we show how to delegate a product of (fixed-base) exponentiations, in a large class of cyclic groups.

Problem History and Related Work

- The **problem** of outsourcing exponentiation to a single malicious server was formally defined and posed in [HL'05] (and studied even earlier), where they gave protocols in the case of **2 servers** of which at most one was malicious, and to **1 server, who was honest on almost all inputs**.

Problem History and Related Work

- The **problem** of outsourcing exponentiation to a single malicious server was formally defined and posed in [HL'05] (and studied even earlier), where they gave protocols in the case of **2 servers** of which at most one was malicious, and to **1 server, who was honest on almost all inputs**.
- Several other solutions either only consider
  - a **semi-honest server**, or
  - **2 non-colluding servers**, or
  - **do not target input privacy**, or
  - **only achieve constant security probability**

Problem History and Related Work

- The **problem** of outsourcing exponentiation to a single malicious server was formally defined and posed in [HL'05] (and studied even earlier), where they gave protocols in the case of **2 servers** of which at most one was malicious, and to **1 server, who was honest on almost all inputs**.
- Several other solutions either only consider
  - a **semi-honest server**, or
  - **2 non-colluding servers**, or
  - **do not target input privacy**, or
  - **only achieve constant security probability**
- Several other solutions consider
  - delegation of **general functions**, or
  - delegation of **linear algebra functions**.

Problem History and Related Work

- The **problem** of outsourcing exponentiation to a single malicious server was formally defined and posed in [HL'05] (and studied even earlier), where they gave protocols in the case of **2 servers** of which at most one was malicious, and to **1 server, who was honest on almost all inputs**.
- Several other solutions either only consider
  - a **semi-honest server**, or
  - **2 non-colluding servers**, or
  - **do not target input privacy**, or
  - **only achieve constant security probability**
- Several other solutions consider
  - delegation of **general functions**, or
  - delegation of **linear algebra functions**.
- Closest result is our previous [DKKS'17] paper where we solve the above open problem for the delegation of a **single** fixed-base exponentiation in a large class of cyclic groups.

Our Contribution

- Consider natural question, motivated by [HL'05] and encryption/signature literature, of whether we can efficiently delegate the **product of multiple exponentiations**.

Our Contribution

- Consider natural question, motivated by [HL'05] and encryption/signature literature, of whether we can efficiently delegate the **product of multiple exponentiations**.
- We show **a protocol** for the delegation to **a single (malicious) server** of a product of fixed-base exponentiations in a large class of cyclic groups
  - This improves the client's number of group multiplications by a factor of about σ with respect to non-delegated computation and a factor of about m with respect to direct use of [DKKS'17].

Our Contribution

- Consider natural question, motivated by [HL'05] and encryption/signature literature, of whether we can efficiently delegate the **product of multiple exponentiations**.
- We show **a protocol** for the delegation to **a single (malicious) server** of a product of fixed-base exponentiations in a large class of cyclic groups
  - This improves the client's number of group multiplications by a factor of about σ with respect to non-delegated computation and a factor of about m with respect to direct use of [DKKS'17].
- We use this result to delegate the **first cryptographic schemes**: the well-known digital signature schemes by El-Gamal, Schnorr and Okamoto. Previously, only primitive operations were delegated.

Our Contribution

- Consider natural question, motivated by [HL'05] and encryption/signature literature, of whether we can efficiently delegate the **product of multiple exponentiations**.
- We show **a protocol** for the delegation to **a single (malicious) server** of a product of fixed-base exponentiations in a large class of cyclic groups
  - This improves the client's number of group multiplications by a factor of about σ with respect to non-delegated computation and a factor of about m with respect to direct use of [DKKS'17].
- We use this result to delegate the **first cryptographic schemes**: the well-known digital signature schemes by El-Gamal, Schnorr and Okamoto. Previously, only primitive operations were delegated.
- In the process, we formally define delegation of digital signature schemes and prove a **conversion theorem** showing that a non-delegated to delegated signature scheme conversion using a suitable delegation protocol for a desired primitive operation

Delegation Protocols: Participant and Interaction Model

Offline phase

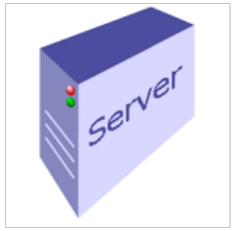
run by Client Deployer or Client, resulting in data stored on Client's device

Online phase

C(x, offline data)

Client's message

Server's message



S

Return: y=F(x)

Delegation Protocols: Algorithm Syntax and Requirements

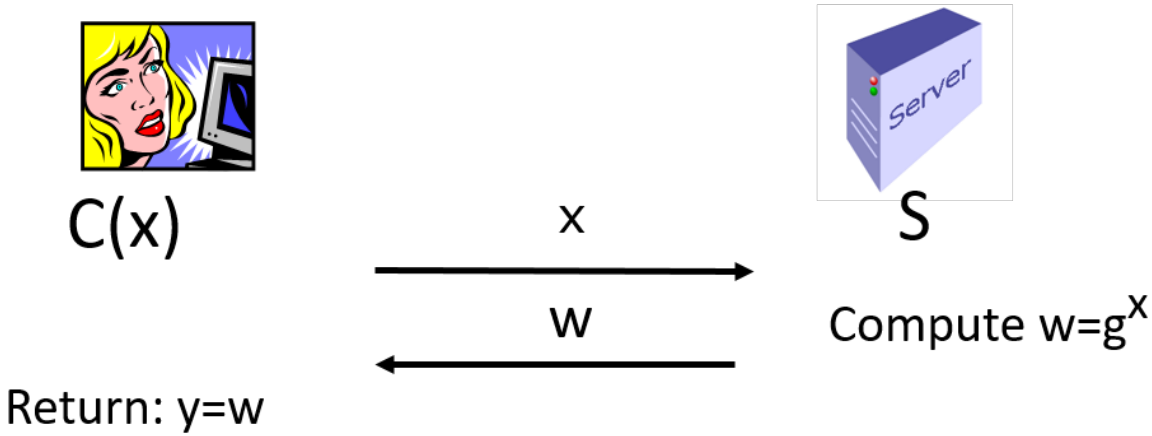
- Model based on [DKKS'17], in turn based on [HL'05] and [GGP'10]
- **Input to client C and server S:** σ (computational parameter), λ statistical parameter, $desc(F)$ (description of delegated function F)
- **Input to C:** x
- **C's output:** $F(x)$, or $\perp$ (failure symbol)
- **Requirements:**
  - **Correctness:** If both parties are honest, C obtains some output at the end of the protocol, and this output should be, with high probability, $= F(x)$, the value obtained by evaluating function F on C's input.
  - **Privacy:** minimal or no information about x should be revealed to S
  - **Security:** a malicious S should not be able, except with very small probability, to convince C to output a non-failure result $\neq F(x)$
  - **Efficiency:** C's computation time should be much smaller than computing $F(x)$ without delegating computation; also runtime of S, runtime in offline mode, round and communication complexity should be kept small.
- Offline computation (by C or others) is allowed before C is given x

Our Delegated Function and Efficiency Requirements

- 1^σ is a computational security parameter (e.g., $\sigma = 2048$)
- 1^λ is a statistical security parameter (e.g., $\lambda = 128$)
- $F_{g_1,\ldots,g_m}(x_1,\ldots,x_m) = y$ where $y = \prod_{i=1}^{m} g_i^{x_i}$ such that
  - $g_1, \ldots, g_m$ are generators of cyclic group $(G, *)$
  - $|g_i| = \sigma$, for $i = 1, \ldots, m$
- Note that $\text{feExp}_{n,e}$ can be evaluated in
  - $\leq 2m\sigma + m - 1$ multiplications, using square-and-multiply algorithm
  - $\leq \frac{2m\sigma}{\log(m\sigma)} + \sigma +$ o-factor multiplications, using improved 'multi-exponentiation' algorithms, including Pippenger's 1976 algorithm
- Main efficiency goals for a protocol (C, S) for a delegated computation of function $F_{g_1,\ldots,g_m}$:
  - C's group multiplications $\ll m\sigma$ and
  - t_S is not significantly larger than t_F.

Towards delegating 1 exponentiation: simplest attempt

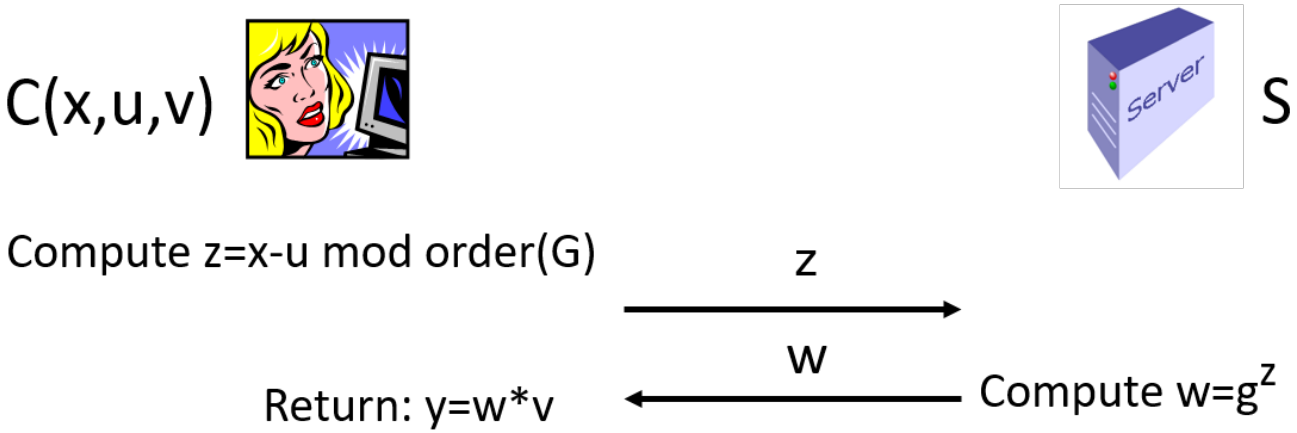
Let G be a cyclic group with generator g known to both C and S



- *Correctness* holds: C obtains $w = g^x$ (here, S is honest)
- *Privacy* is not satisfied: S learns x
- *Security* is not satisfied: C cannot trust its result y (here, S may not be honest)
  - Note: C is computationally weak and cannot compute g^x
  - this also rules out using proofs of knowledge of a dlog exponent

Towards delegating 1 exponentiation: input masking

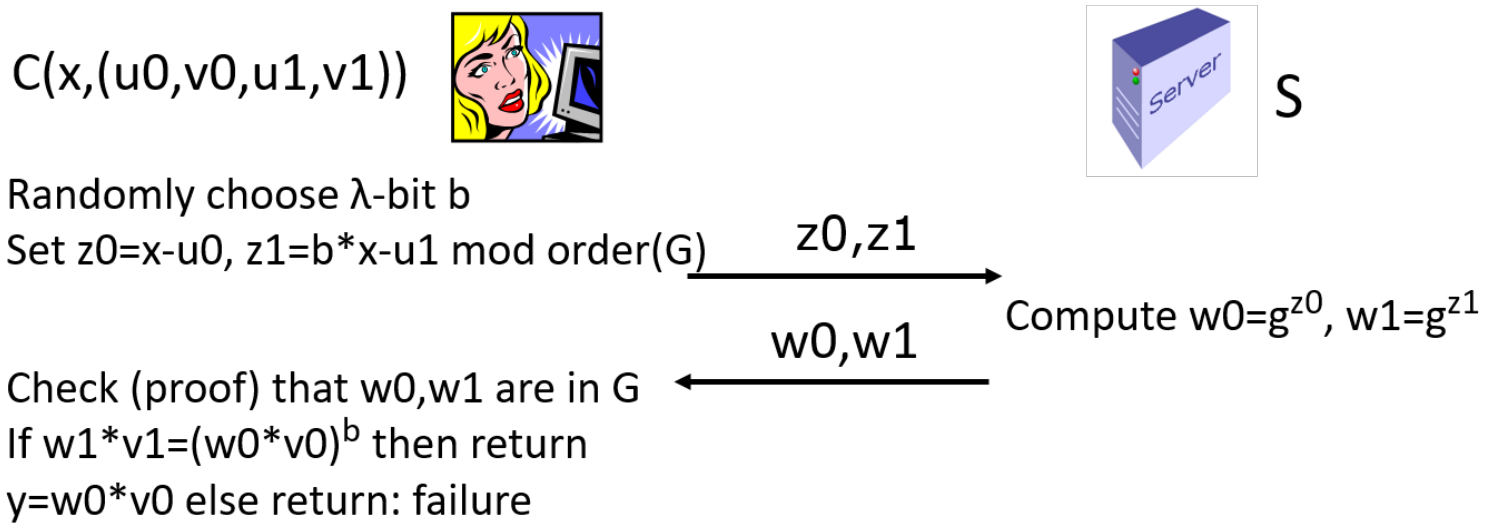
Let G be a cyclic group with generator g known to both C and S. Someone (Deployer or Client) precomputes (u,v), where u is random in {1,..,order(G)} and $v=g^u$, and stores (u,v) on Client.



- *Correctness* holds: C obtains $w = g^x$ (here, S is honest)
- *Privacy* is satisfied: no S can learn any information about x
  - Input masking gives us privacy
- *Security* is still not satisfied: C cannot trust its result y (here, S may not be honest)
- Assuming order(G) is known to C

Protocol from [DKKS17] to delegate 1 exponentiation

Deployer or Client precomputes (u[j],v[j]), where u[j] is random in {1,..,order(G)} and $v[j]=g^{u[j]}$, and stores (u[j],v[j]), j=0,1, on Client

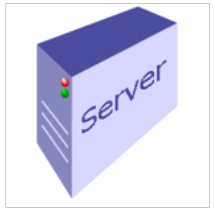


- *Ingredients*: input masking, small-coefficient linear test, efficient group membership test
- Satisfies *Correctness*, *Privacy*, *Security* and *Efficiency*; C performs $\leq 2\lambda + 3$ multiplications (e.g., 259); compare with $\leq 2\sigma$ (e.g., 4096)
- *Group assumptions*: cyclic, g known, efficient group membership test

Using [DKKS17] protocol for a product of exponentiations

Deployer or Client precomputes (u[j],v[j]), where u[j] is random in {1,..,order(G)} and $v[j]=g^{u[j]}$, and stores (u[j],v[j]), j=0,1, on Client

C(x[1],...,x[m]) 



S

Randomly choose λ-bit b[1],...,b[m]
Set z[0i]=x+u[0i], z[1i]=b[i]*x+u[1i]

z[0i],z[1i] →

← w[0i],w[1i]

Compute $w0i=g^{z0i}$, $w1i=g^{z1i}$

Check that w[0i],w[1i] are in G
If $w[1i]/v[1i]=(w[0i]/v[0i])^{b[i]}$ then compute
y[i]=w[01]/v[01] and return y=y[1]*...*y[m]
else return: failure

- Essentially a parallel repetition of [DKKS17]
- C does $\leq 2\lambda m + 4m - 1 << \sigma m$ multiplications (linear in λm)

Ideas behind our new protocol

- Peeliminary discussion:
  - In previous protocol gap between S′s runtime and Cs runtime becomes less practically relevant as m gets large; but products of exponentiations are frequent in crypto protocols. Can we do better?
  - Can C verify m simultaneous exponentiations faster than independently verifying all m of them? No, or not much faster
  - Does C need to compute m simultaneous exponentiations? No, only a product of m exponentiations
- Ideas behind our new protocol:
  - Input masking on each of the m exponentiations
  - Small-coefficient linear test is now run on the product of exponentiations instead of on the single exponentiation; can use a single test (instead of m)
  - Let S and offline deployer compute products of exponentiations (instead of C)

Our new protocol for a product of exponentiations

Deployer or Client precomputes (u[ij],v[ij]), where u[ij] is random in {1,..,order(G)} and v[j]= $\prod g^{u[ij]}$, and stores (u[1j],...,u[mj],v[j]), j=0,1, on Client

C(x[1],...,x[m]) 

S

Randomly choose λ-bit b
Set z[0i]=x+u[0i], z[1i]=b*x+u[1i]

z[0i],z[1i] →

← w0,w1

Check that w0,w1 are in G
If w1/v1=(w0/v0)^b then return y=w0/v0 else return: failure

Compute
w0= $\prod g^{z[0i]}$
w1= $\prod g^{z[1i]}$

- *Ingredients*: masking m inputs, single small-coefficient linear test, only 2 efficient group membership tests
- Provable correctness, privacy, $2^{-\lambda} - security$, efficiency
- C does $\leq 2\lambda + 3m << \sigma m$ multiplications (linear in $\lambda + m$)

Analysis of Client Runtime

Table 1. Comparison of number of C's online multiplications in $\mathbb{Z}_p^*$ where $p = 2q + 1$

	m	$F_{g_1,\ldots,g_m,q}$ No delegation		$F_{g_1,\ldots,g_m,q}$ With delegation	
		nPoExp	*fPoExp*	*nDelPoExp*	Our result
$\sigma = 2048$ $\lambda = 128$	2	8,193	6,144	520	261
	5	20,484	12,288	1,300	264
	10	40,969	22,528	2,600	269
	50	204,849	104,448	13,000	309
	100	409,699	206,848	26,000	359
	1000	4,096,999	2,050,048	260,000	1,259
	Arbitrary m	$4,097m - 1$	$2,048(m + 1)$	$260m$	$259 + m$
Arbitrary m, σ, λ		$2m\sigma + m - 1$	$m\sigma + \sigma$	$2m\lambda + 4m$	$2\lambda + m + 3$

- *Main Takeaways*:
  - Improvement with respect to undelegated computation is a multiplicative factor of about σ
  - Improvement with respect to parallel use of [DKKS17] delegation of single exponentiation is a multiplicative factor of about m

Applications: Delegation to Signature Schemes

1. Main result: we show efficient, private and secure delegation schemes for well-known (i.e., ElGamal, Schnorr and Okamoto's) signature schemes using the delegation of a product of (fixed-base) exponentiation for cyclic groups
2. We augment the unforgeability definition of (non-delegated) signature scheme so to additionally take into account eavesdropping and oracle query attacks in the delegated model.
3. We present a general result that shows how to convert signature schemes in the non-delegated model into signature schemes in the delegated model by using a suitable delegation protocol.
4. Finally, we show delegated ElGamal, Schnorr and Okamoto's signature schemes.

Non-Delegated Signature Schemes: definitions review

- 3 Algorithms:
  - Key generation returns Alices pk and sk
  - With Sign, Alice can compute a signature sig for message m using pk,sk,
  - With Verify, Bob can verify (m,sig) using Alices pk
- 2 requirements:
  - Correctness: Bob can verify Alices (mes,sig) pair based on Alices pk
  - Unforgeability: Even after polynomial adaptive queries to the Sign(pk,.) oracle, no efficient adversary can forge a signature for a new message, unless with negligible probability

Delegated Signature Schemes: definitions

- *Syntax* of delegated signature schemes:
  - signer and verifier are oracle algorithms that may use, as an oracle, the delegation server
  - both the signer and the verifier can act as client in a (C,S) instance
- *Correctness* is a direct extension of the non-delegated case
- For *unforgeability*, we replace the adversary A's oracle $Sign(pk, sk, \cdot)$ with two oracles:
  - an augmented oracle $dSign(pk; sk; \cdot)$ which, on input message m, returns a signature *sig* and the transcript of any query/answer interaction with the server oracle S performed by Sign during generation of *sig*; this models *chosen-message* and *eavesdropping attacks*;
  - the server oracle $S(desc(F); \cdot)$, which, on input C's query message, returns S's response to this message in an execution of protocol (C,S); this models *collusion with a client or a server*.

Delegated Signatures: results

- (Informal) General Theorem: Given a secure signature scheme where signer and/or verifier use F and and a delegation protocol for F we can construct a secure delegated signature scheme where signer and/or verifier delegates F
  - Note: Privacy of delegation protocol has to be based on a simulation-based definition, which we achieve
  - Consequence: Can replace any computation by signers and verifiers with its delegation and keep security
  - We replace the most expensive step in the verifier's computation of a product of 2 or 3 exponentiations in the well-known signature schemes by ElGamal, Schnoor and Okamoto, using our new delegation protocol
    - The number of online group multiplications by the verifier is reduced by a factor between 20 and 40, and remains less than 260 in the worst case

Conclusions

- We solved the problem of delegating the computation of a product of group exponentiations to a single, possibly malicious, server, in a large class of cyclic groups; specifically, cyclic groups whose multiplication and inverse operations can be efficiently computed, and which admit an efficiently verifiable protocol to prove that an element is in the group. The considered class of cyclic groups includes groups often discussed in cryptography literature, such as prime-order subgroups in Z_p and elliptic curve groups.
  - Future work: working on more groups, including post-quantum ones; p-groups, etc.
- As an application, we showed the first delegation of an entire cryptographic scheme. Previous research only delegated a single operation of a scheme's algorithm.
  - Future work: other, more general, schemes?
- Open problem: This paper, and our previous 2 papers are on delegation of fixed-base exponentiation. Private, secure and efficient delegation of (even single) variable-base exponentiation is open.

Thank You!

Questions?

References


A. Arbit and Y. Livne and Y. Oren and A. Wool, *Implementing public-key cryptography on passive RFID tags is practical.* In: Int. J. Inf. Sec. 14(1): pp. 85-99, 2015.


M. Atallah, K. Pantazopoulos, J. Rice, E. Spafford, *Secure outsourcing of scientific computations*. In Adv. Comput. 54, pp. 215–272, 2002.


M. Atallah and K. Frikken, *Securely outsourcing linear algebra computations*, In. Proc. of 5th ACM ASIACCS, 2010, pp. 48–59.


P. Barrett, *Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor*, In Proc. of CRYPTO 1986, LNCS 263, pp. 311–323, 1986.


L. Batina and J. Guajardo and T. Kerins and N. Mentens and P. Tuyls and I. Verbauwhede, *Public-Key Cryptography for RFID-Tags.* In: 5th IEEE International Conference on Pervasive Computing and Communications - Workshops (PerCom Workshops 2007), pp. 217–222, 2007.


M. Bellare, J. Garay, and T. Rabin, *Fast batch verification for modular exponentiation and digital signatures.* In Proc. of Eurocrypt 1998: pp. 236–250, Springer, 1998.


A. Brauer, *On Addition Chains*, Bulletin of the American Mathematical Society, vol. 45, pp. 736–739, 1939


D. Benjamin and M. Atallah, *Private and cheating-free outsourcing of algebraic computations*, In Proc. of 6th PST 2008, Springer-Verlag, pp. 240245.


V. Boyko and M. Peinado and R. Venkatesan, *Speeding up discrete log and factoring based schemes via precomputations.* In Proc. of EUROCRYPT'98, pp. 221–235, Springer, 1998.


E. Brickell, D. Gordon, Z. Mccurley, and D. Wilson, *Fast exponentiation with precomputation*. In Proc. of Eurocrypt 92, LNCS Vol. 658, Springer-Verlag, 1992.

References


J. Cai, Y. Ren, and T. Jiang, *Verifiable Outsourcing Computation of Modular Exponentiations with Single Server*, In: International Journal of Network Security, 19 (3), pp. 449–457, (2017)


B. Cavallo, G. Di Crescenzo, D. Kahrobaei, and V. Shpilrain, *Efficient and secure delegation of group exponentiation to a single server*, In: International Workshop on Radio Frequency Identification: Security and Privacy Issues: pp. 156–173, Springer, 2015.


X. Chen and J. Li and J. Ma and Q. Tang and W. Lou, *New algorithms for secure outsourcing of modular exponentiations.* In: Computer Security–ESORICS 2012, pp. 541-556, 2012.


C. Chevalier, F. Laguillaumie, D. Vergnaud, *Privately Outsourcing Exponentiation to a Single Server: Cryptanalysis and Optimal Constructions* In Proc. of ESORICS 2016: pp. 261-278


K. Chung K, Y. Kalai, and S. Vadhan, *Improved delegation of computation using fully homomorphic encryption*. In Proc. of 30th Annual Cryptology Conference, Santa Barbara, CA, USA, in: Lect. Notes Comput. Sci., vol. 6223, Springer-Verlag, August 2010, pp. 483501.


R. Cramer, Victor Shoup, *Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack*. In SIAM J. Comput. 33(1): 167-226 (2003)


B. Cubaleska, A. Rieke, and T. Hermann, *Improving and extending the Lim/Lee exponentiation algorithm*, In Proc. of International Workshop on Selected Areas in Cryptography, pp. 163–174, Springer (1999)


G. Di Crescenzo, D. Kahrobaei, M. Khodjaeva, V. Shpilrain, *Efficient and Secure Delegation to a Single Malicious Server: Exponentiation over Non-abelian Groups*. In Proc. of ICMS 2018: pp. 137–146


G. Di Crescenzo, M. Khodjaeva, D. Kahrobaei, and V. Shpilrain, *Computing Multiple Exponentiations in Discre Log and RSA Groups: From Batch Verification to Batch Delegation*, In: Proc. of 3rd IEEE Workshop on Security and Privacy in the Cloud, IEEE, 2017.

References


G. Di Crescenzo, M. Khodjaeva, D. Kahrobaei, and V. Shpilrain, *Practical and Secure Outsourcing of Discrete Log Group Exponentiation to a Single Malicious Server*. In Proc. of 9th ACM Cloud Computing Security Workshop (CCSW), pp. 17-28, 2017


W. Diffie, M. E. Hellman, *New directions in cryptography*. In IEEE Transactions on Information Theory 22(6): 644-654 (1976)


M. Dijk, D. Clarke, B. Gassend, G. Suh, and S. Devadas, *Speeding Up Exponentiation using an Untrusted Computational Resource*. In: Designs, Codes and Cryptography, 39 (2), pp. 253-273, 2006.


V. Dimitrov, G. Jullien, W. Miller, *An Algorithm for Modular Exponentiation*. Inf. Process. Lett. 66(3): 155-159 (1998)


Y. Ding, Z. Xu, J. Ye, and K. Choo, *Secure outsourcing of modular exponentiations under single untrusted programme model*. In Journal of Computer and System Sciences, vol.90, C, Academic Press, Inc., pp. 1-13, 2017.


T. El Gamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*. In IEEE Transactions on Information Theory 31(4): 469-472 (1985)


D. Fiore and R. Gennaro, *Publicly verifiable delegation of large polynomials and matrix computations, with applications*. In Proc. of ACM CCS Conference 2012, pp. 501–512.


R. Gennaro, C. Gentry, and B. Parno, *Non-interactive verifiable computing: Outsourcing computation to untrusted workers,* in Proc. of CRYPTO 2010, LNCS 6223, pp. 465–482, 2010.


C. Gentry, *Fully homomorphic encryption using ideal lattices*. In Proc. of STOC 09, 2009, pp. 169178.


S. Hohenberger and A. Lysyanskaya, *How to securely outsource cryptographic computations.* In: Proceedings of the Theory of Cryptography Conference 2005, pages 264-282, Springer, 2005.

References


M. Jakobsson and S. Wetzel, *Secure server-aided signature generation.* In: Proceedings of the Public Key Cryptography conference, pp. 383-401, Springer, 2001.


C. Lim, and P. Lee, *More flexible exponentiation with precomputation*, In: Proceedings of CRYPTO 1994, pages 95-107.


X. Ma and J. Li and F. Zhang, *Outsourcing computation of modular exponentiations in cloud computing.* In: Cluster Computing (2013) 16:787-796 (also INCoS 2012).


T. Matsumoto, K. Kato and H. Imai, *An improved algorithm for secure outsourcing of modular exponentiations*. In Proc. of CRYPTO 1988, pp. 497506.


B. Möller, *Improved techniques for fast exponentiation*, In: Proceedings of ICISC, (2587): pp. 298-312, Springer (2002)


P. Q. Nguyen and I. E. Shparlinski and J. Stern, *Distribution of modular sums and the security of the server aided exponentiation.* In: Proceedings of Cryptography and Computational Number Theory, pp. 331-342, Springer, 20


T. Okamoto *Provably secure and practical identification schemes and corresponding signature schemes*. In Proc. of CRYPTO 1992, pp. 31-53.


C. Schnorr, *Efficient Signature Generation by Smart Cards*, in Journal of Cryptology 4(3), pp. 161–174, 1991


E. Straus, *Addition Chains of Vectors (problem 5125)*, American Mathematical Monthly, vol. 70, (1973), pp. 907-913


Y. Wang and Q. Wu and D. Wong and B. Qin and S. Chow and Z. Liu and X. Tao, *Securely outsourcing exponentiations with single untrusted program for cloud storage.* In: Proceedings of Computer Security-ESORICS 2014, pp.326-343.


A. C. Yao, *Protocols for secure computations.* In: Proceedings of 23rd FOCS, pp. 160-168, IEEE, 1982.


L. Zhao, M. Zhang, H. Shen, Y. Zhang, and J. Shen, *Privacy-preserving Outsourcing Schemes of Modular Exponentiations Using Single Untrusted Cloud Server*. In: KSII Transactions on Internet & Information Systems 11 (2)