CSAIL

Computer Science and Artificial Intelligence Laboratory

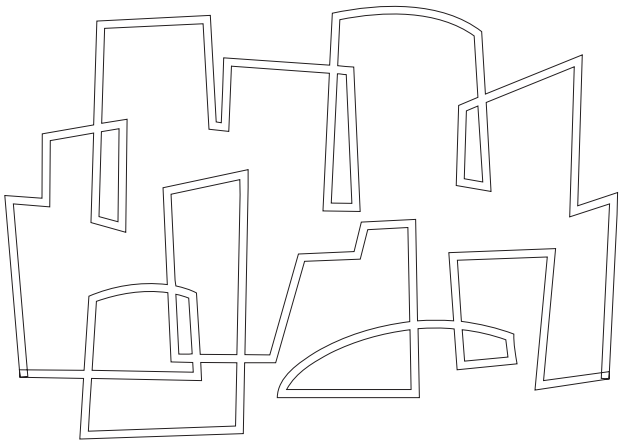
Massachusetts Institute of Technology

Speeding up Exponentiation using an Untrusted Computational Resource

Dwaine Clarke, Srinivas Devadas, Marten van Dijk,
Blaise Gassend, G. Edward Suh

2003, August

Computation Structures Group
Memo 469



The Stata Center, 32 Vassar Street, Cambridge, Massachusetts 02139

Speeding up Exponentiation using an Untrusted Computational Resource

Dwaine Clarke*, Srinivas Devadas, Marten van Dijk**,
Blaise Gassend, G. Edward Suh

MIT Computer Science and Artificial Intelligence Laboratory
{declarke, devadas, marten, gassend, suh}@mit.edu

August 28th, 2003

Abstract. We present protocols for speeding up fixed-base exponentiation and variable-base exponentiation using an untrusted computation resource. In the fixed-base protocols, the base and exponent may be blinded. If the exponent is fixed, the base may be blinded in the variable-base exponentiation protocols. The protocols are the first ones for accelerating exponentiation with the aid of an untrusted resource in arbitrary cyclic groups. We also describe how to use the protocols to construct protocols that do, with the aid of an untrusted resource, exponentiation modular an integer where the modulus is the product of primes with single multiplicity.

One application of the protocols is to speed up exponentiation-based verification in discrete log-based signature and credential schemes. For example, the protocols can be applied to speeding up, on small devices, the verification of signatures in DSS, El Gamal, and Schnorr's signature schemes, and the verification of digital credentials in Brands' credential system.

The protocols use precomputation and we prove that they are unconditionally secure. We analyze the performance of our variable base protocols where the exponentiation is modulo a prime p: the protocols provide an asymptotic speedup of about $\mathcal{O}(0.24(\frac{k}{\log k})^{\frac{2}{3}})$, where $k = \log p$, over the square-and-multiply algorithm, without compromising security.

1 Introduction

We study the general theoretical problem of speeding up computations using an untrusted computational resource. In particular, we introduce a few protocols to speed up exponentiation. The schemes are unconditionally secure, and do not rely on any hardness assumptions.

We also introduce a general paradigm for designing protocols addressing the problem of checking the integrity of untrusted computation. The paradigm allows simpler protocols to be used as building blocks to build more complex, multistage protocols. In earlier stages of the multistage protocol, the untrusted resource commits to various values. Later, the trusted resource uses the untrusted resource to test the integrity of the committed values. We term this paradigm, the *commit-and-test* paradigm.

Much of the previous work trying to speed up exponentiation using an untrusted resource, e.g. [MKI89,KS93,BQ95], has been focused on trying to speed up RSA [RSA78]

* Note: authors are listed alphabetically.

** Visiting researcher from Philips Research, Prof Holstlaan 4, Eindhoven, The Netherlands.

computations. The protocols do variable-base, fixed-exponent exponentiation while trying to keep the exponent secret from the untrusted resource. However, these schemes have been shown to be insecure [PW93,And92,NS98]. We do not try to solve this problem. Instead, we introduce provably secure protocols. In particular, we introduce four protocols, two fixed-base exponentiation protocols and two variable-base exponentiation protocols. We present a protocol for fixed-base exponentiation, and then present a second protocol which has a logarithmic security improvement over the first one. In these protocols, the base and exponent may be blinded. Using the commit-and-test paradigm, we then use the fixed-base protocols to build a protocol for variable-base exponentiation. The fourth protocol is a variable-base protocol showing that, if the exponent is fixed, the base may blinded. We also describe how to use the protocols to construct protocols that do, with the aid of an untrusted resource, exponentiation modular an integer where the modulus is the product of primes with single multiplicity.

The problem of efficiently, and provably, checking the integrity of untrusted computation is an interesting theoretical problem in its own right. In [BK89] paper, Blum and Kannan introduced the concept of a program checker, an algorithm that checks the output of a particular computation. Some of the checkers in [BK89], such as the checkers for sorting and *gcd* (greatest common divisor) programs, can be run by a trusted resource to check the results of an untrusted resource. In some sense, the schemes we describe in this paper are 'program checkers' for the exponentiation algorithm.

Applications of our variable base exponentiation protocol include speeding up signature *verification* of several public key-based signature schemes. There are several signature schemes, such as DSS [NIS94], El Gamal's signature scheme [ElG85] and Schnorr's [Sch91] signature scheme, for which the online generation of message signatures can be done quickly because the part of the signature that requires exponentiation can be precomputed. However, signature verification, in general, requires online exponentiation because the message signatures are used in the exponentiation. Our protocols enable the speedup of signature verification where exponentiation is modulo a prime. A small device, such as a handheld or sensor, can use the protocol to verify signatures using the computation of a powerful, untrusted desktop computer or server. Because the untrusted resource may be compromised, and may return false values, the handheld's owner will not wish to trust it. He would like the trusted resource to verify the untrusted resource's computation. We analyze the performance of our variable base protocols where the exponentiation is modulo a prime p, and conclude that they provide an asymptotic speedup of about $\mathcal{O}(0.24(\frac{k}{\log k})^{\frac{2}{3}})$, where $k = \log p$ (i.e. k is the number of bits used to represent p), over the square-and-multiply algorithm, without compromising security.

1.1 Our Main Contributions

The protocols are the first ones for accelerating exponentiation with the aid of an untrusted resource in arbitrary cyclic groups. We also describe how to use the group exponentiation protocols to construct protocols to do exponentiation modular an integer where the modulus is the product of primes with single multiplicity. Another significant contribution is demonstrating how the commit-and-test paradigm can be used to design protocols which address the untrusted computation problem.

1.2 Organization

The paper is organized as follows. Our model is described in Section 2. Section 3 details our protocols: Section 3.1 describes the two fixed-base protocols; Section 3.2 describes the two variable-base protocols; the protocols are described for exponentiation in arbitrary cyclic groups, and Section 3.3 describes how to use them to construct protocols for modular exponentiation; Section 3.4 discusses the precomputation phase of the protocols.

Section 4 describes applications for our protocols, and relates the performance of the protocols to the security in these applications. Section 5 describes related work, and Section 6 concludes the work. The appendices prove the security of our protocols.

2 Model

The model is simple. Tim, a trusted device, is given an input P, representing a problem that he would like to solve. It wishes to leverage the computational power of a more powerful, but untrusted, server, Ursula. Because, Ursula is untrusted, Tim needs to check the integrity of the results it receives from Ursula.

In our model, we allow multiple rounds of communication between Tim and Ursula. We also allow Tim to perform precomputations in its idle time. We refer to *precomputation* as the work Tim performs before it gets input P to solve; the work Tim performs after it gets P is referred to as *online work*. For a comparable security factor, the online work Tim performs must be less than the work Tim would have performed if it had solved P itself. In essence, our schemes are able to offload the work Tim performs to a precomputation stage, such that the online work it performs when it gets a problem to solve is very small. Our model is fairly relaxed, but, in it, we are able to introduce schemes that address the problem of checking the integrity of untrusted computation, that are provably secure, and for which there is a very significant speedup.

We do not explicitly include Tim's communication costs in our model, as this depends on the particular application scenario. For example, for a handheld in a dock at an untrusted desktop computer, which is trying to leverage the computational power of the desktop computer [BDF+01], the communication costs can be very small. We do note that the speedup of our protocols is significant, and, thus, if Tim's communication costs are sufficiently small, Tim's online work including its communication costs when it uses our protocol will be smaller than if it had solved the problem itself.

3 Exponentiation

Suppose that Tim wants to compute the exponentiation g^a within a cyclic group G for some base $g \in G$ and exponent a. In general such computation costs $\mathcal{O}(\log n)$ multiplications, where n is the order of the cyclic group ($g^n = 1$ for all $g \in G$). Tim's goal is to compute g^a with less multiplications by interacting with Ursula, an untrusted coprocessor who has enough computing power. In this section we describe protocols to do fixed and variable base exponentiation and we describe how these protocols can be applied to do modular exponention (exponentation in an integer ring).

3.1 Fixed Base Exponentiation

We assume that the cyclic group G and the factorization of the order of the cyclic group n are publicly known. Let us consider the protocol in Figure 1 where the security parameter s, $s \leq n$, and derived parameters w_s and q_s satisfy $n = w_s q_s$,

- primes dividing q_s are at least s or equal to s, and
- primes dividing w_s are at most s.

Notice that w_s increases as s increases. We assume that Ursula is untrusted and knows the inputs g and a and parameters $w_s = \lfloor a/e \rfloor$, n, and $q_s = n/w_s$.

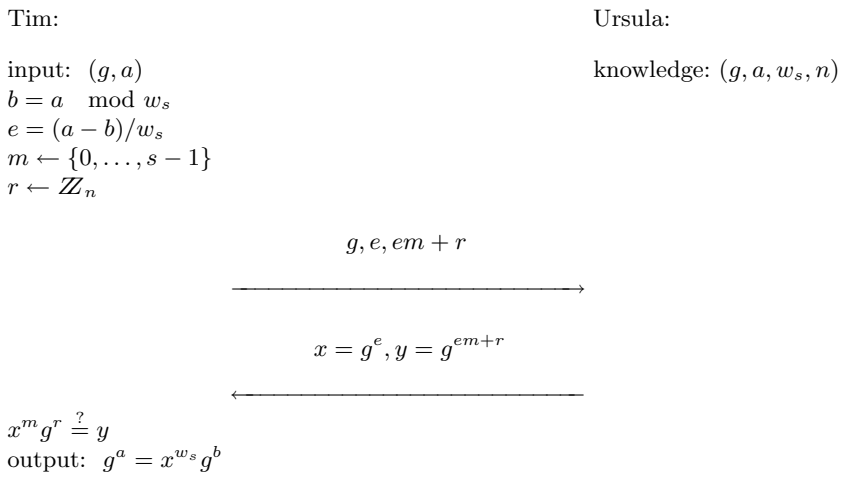


Fig. 1. Fixed base exponentiation using an untrusted coprocessor Ursula. The probability of successful attack is equal to $1/s = 2^{-\log s}$. Tim's precomputation costs $\mathcal{O}(\log n)$ multiplications. Tim's and Ursula's workload during the protocol is $\mathcal{O}(\log s + \log w_s)$ and $\mathcal{O}(\log n)$ multiplications respectively. By using an extra blinding factor the exponent a can be kept secret from Ursula.

Theorem 1. *For the protocol depicted in Figure 1, the probability of successful attack, that is the probability that Ursula is able to tamper with x and y such that Tim outputs an incorrect value without detecting Ursula's tampering, is equal to $1/s$.*

The proof is in Appendix A.

The protocol in Figure 1 shows that besides some multiplications Tim needs to compute x^m, g^r, x^{w_s}, and g^b. If the base g is fixed and known beforehand, the exponentiation g^r can be done during a precomputation. If g is variable, this protocol is of less use since Tim is better off computing g^a all by himself. So, for a fixed base exponentiation Tim only needs to compute x^m, with $m \leq s$, x^{w_s}, and g^b, with $b < w_s$, during the protocol. This costs $\mathcal{O}(\log s + \log w_s)$ multiplications. Notice that Ursula performs two simultaneous exponentiations in the cyclic group, which costs $\mathcal{O}(\log n)$ multiplications.

The value g^r is called a blinding factor because it hides m from Ursula. For cryptographic reasons we may introduce a second blinding factor to hide the exponent e from Ursula, such that she does not get any knowledge about the input a. Then instead of e Tim sends $e + r'$ to Ursula, who computes $x' = g^{e+r'}$, after which Tim computes $x = x'/g^{r'}$.

At the cost of extra computations for Ursula and Tim, we may introduce additional checks which decrease the probability of successful attack by Ursula. The protocol of Figure 2 implements the protocol of Figure 1, with security parameter s', t times in parallel. It follows from Theorem 1 that Ursula's probability of successful attack is equal to $1/s'^t$.

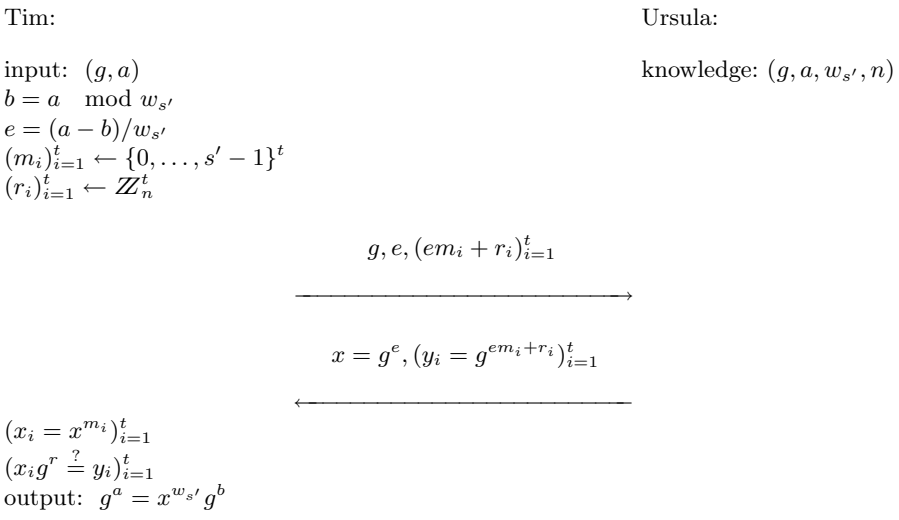


Fig. 2. A general protocol for fixed base exponentiation using an untrusted coprocessor Ursula. The probability of successful attack is equal to $1/s'^t = 2^{-t \log s'}$. Tim's precomputation costs $\mathcal{O}(t \log n)$ multiplications. Tim's and Ursula's workload during the protocol is $\mathcal{O}(\min\{s', t \log s'\} + \log w_{s'})$ and $\mathcal{O}(t \log n)$ multiplications respectively. By using an extra blinding factor the exponent a can be kept secret from Ursula.

Since the original protocol is implemented t times in parallel Tim's precomputations and Ursula's workload during the protocol are increased by a factor t. At first sight Tim's workload during the protocol is increased by a factor t as well. However, during the protocol Tim only needs to compute $(x^{m_i})_{i=1}^t$, with each $m_i \leq s'$, $x^{w_{s'}}$, and g^b, with $b < w_{s'}$. To compute $(x^{m_i})_{i=1}^t$, Tim may simply compute all possible values for x^{m_i}, that is x^j, for $0 \leq j \leq s' - 1$. This costs only $\mathcal{O}(s')$ multiplications. For example, take $s' = \log s$ and $t = \log s$ then the probability of successful attack is equal to $2^{-\log s \log \log s} < 2^{-\log s}$ and compared to Figure 1 Tim's precomputation and Ursula's workload during the protocol are increased by a factor $\log s$, while Tim's workload during the protocol remains the same.

3.2 Variable Base Exponentiation

Figure 3 depicts a protocol in which Tim uses Ursula to compute g^a for a variable base g. Notice that Tim can perform the exponentiation γ^r during a precomputation. This means that during the protocol Tim only needs to compute $g^{m'}$, x^m, $z^{m'}$, with $m, m' < s$, z^{w_s}, and g^b, with $b < w_s$. As in our protocol for fixed base exponentiation this costs $\mathcal{O}(\log s + \log w_s)$ multiplications.

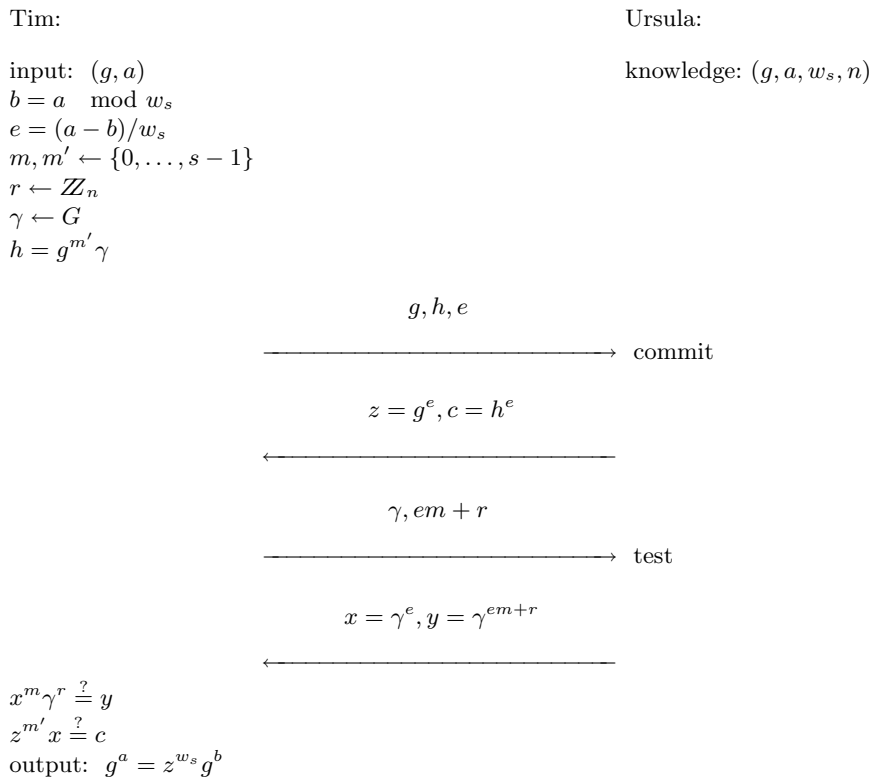


Fig. 3. Variable base exponentiation using an untrusted coprocessor Ursula. The probability of successful attack is equal to $2/s \approx 2^{-\log s}$. Tim's precomputation costs $\mathcal{O}(\log n)$ multiplications. Tim's and Ursula's workload during the protocol is $\mathcal{O}(\log s + \log w_s)$ and $\mathcal{O}(\log n)$ multiplications respectively.

The security is based on the *commit and test* paradigm. We let Ursula compute z and c before she knows γ. This commits $z = g^e$ to an algebraic relationship with c: $z^{m'}\gamma^e = c$. Tim is not performing this check all by himself because this would involve the exponentiation γ^e, which can not be precomputed if e is variable and not fixed beforehand. He lets Ursula do this exponentiation by using the fixed base protocol of Figure 1.

Theorem 2. *For the protocol depicted in Figure 3, the probability of successful attack, that is the probability that Ursula is able to tamper with z, c, x and y such that Tim outputs an incorrect value without detecting Ursula's tampering, is equal to $2/s$.*

The proof is in Appendix B.

If the exponent a is fixed beforehand then Tim can precompute γ^e and the protocol of Figure 3 gets simplified to the one in Figure 4. In the simplified protocol we may introduce a second blinding factor to hide the base g from Ursula. Then instead of g Tim sends $g\gamma'$ to Ursula, who computes $x' = (g\gamma')^e$, after which Tim computes $x = x'/\gamma'^e$. For completeness we mention that we can generalize the simplified protocol by using the technique used in Figure 2.

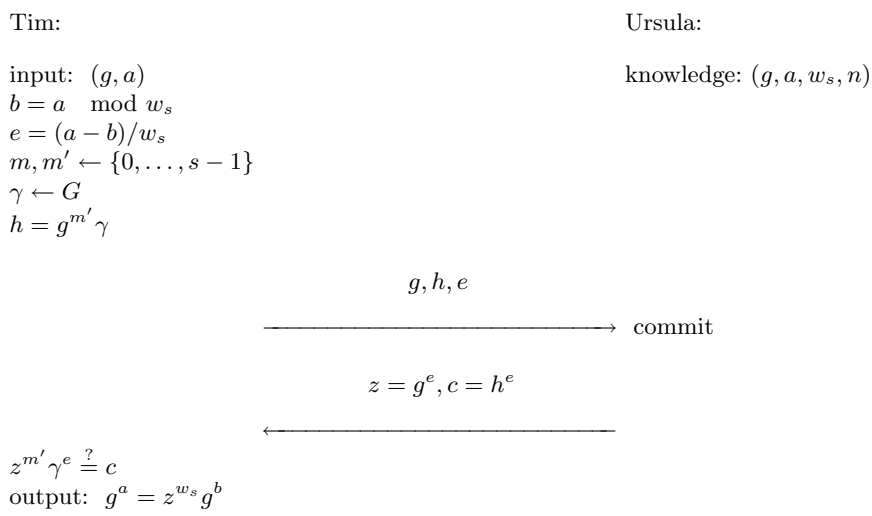


Fig. 4. Variable base with fixed exponent exponentiation using an untrusted coprocessor Ursula. The probability of successful attack is equal to $1/s = 2^{-\log s}$. Tim's precomputation costs $\mathcal{O}(\log n)$ multiplications. Tim's and Ursula's workload during the protocol is $\mathcal{O}(\log s + \log w_s)$ and $\mathcal{O}(\log n)$ multiplications respectively. By using an extra blinding factor the base g can be kept secret from Ursula.

3.3 Modular Exponentiation

Suppose that Tim wants to perform a modular exponentiation, that is he wants to compute g^a modulo an integer n, where $g \in \mathbb{Z}_n$ and $g \neq 0$. As an example we modify the fixed base exponentiation protocol of Figure 1. The first difference is that exponentiations are computed in the ring $\mathbb{Z}_n$, that is we perform integer exponentiation modulo n. Secondly, in the new protocol the security parameter $s \leq \phi(n)$ and w_s and q_s are such that $\phi(n) = w_s q_s$, primes dividing q_s are at least or equal to s, and primes dividing w_s are at most s. Here $\phi(n)$ is the Euler function evaluated in n, that is

$$\phi(n) = \prod_j \phi(p_j^{e_j}) = \prod_j (p_j - 1) p_j^{e_j - 1},$$

where $\prod_j p_j^{e_j}$ is the factorization of n for distinct primes p_j. We recall that Ursula only knows the inputs g and a and the parameters $w_s = \lfloor a/e \rfloor$ and n, which are used in

the protocol. Notice that, even though $w_s|\phi(n)$ these parameters do not directly reveal the factorization of n to Ursula (or Tim). This is important if we wish to apply our protocols in for example the RSA signature scheme where the security is based on hiding the factorization of n. To model into what extend the protocol reveals the factorization of n, we assume that $n = wq$, where

- the factorization of w is possibly known to principals who have access to w, w_s and n, and
- it is infeasible to compute any prime factor of q given knowledge of w, w_s and n.

These assumptions lead to the modified exponentiation protocol with fixed base g of Figure 5, where $gcd(g, w) = 1$. Its security is proved in Appendix C, where we also show how to use this protocol to compute g^a modulo n for a general fixed base g (without the restriction $gcd(g, w) = 1$) but with the restriction that n has only prime factors with single multiplicity (that is, each $e_j = 1$). We leave it to the reader to verify that the other protocols of the previous subsections can be modified in a similar way.

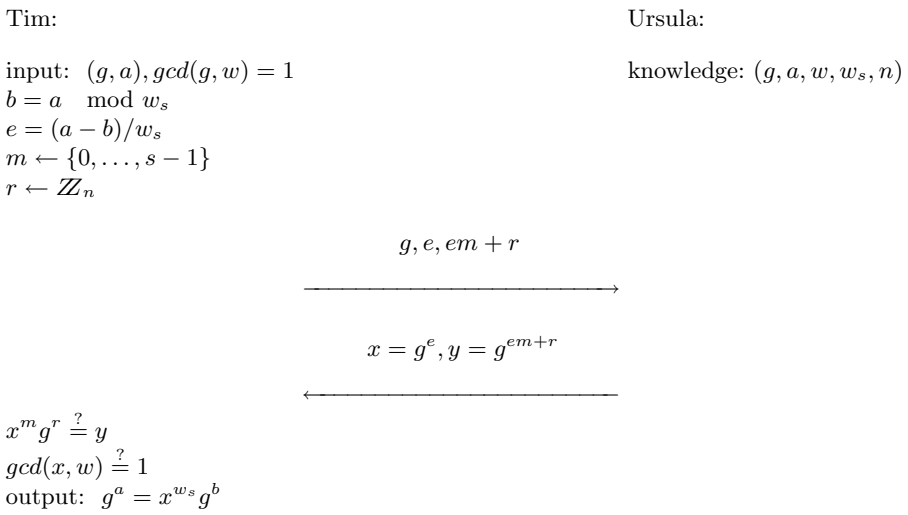


Fig. 5. Protocol to be used for fixed base modular exponentiation using an untrusted coprocessor Ursula. It has the same properties as the protocol in Figure 1.

3.4 Precomputation

In the precomputation phase of the variable base exponentiation protocols, Tim needs to generate triples (r, γ, γ^r) where x and γ are random. We suggest that Tim generate these triples himself for the following reason. It is tempting to try to use more efficient methods to generate the triples. As a brief historical note, in [Sch90], Schnorr suggested a method by which the trusted resource stores a collection of independent random pairs

(r, g^r) where g is fixed. Whenever it needs a pair, it creates a random combination of the pairs in its storage, and uses the result as the pair. The scheme has been shown to not be secure. After the first pair is used, the resultant subsequent pairs are not guaranteed to be random, and each generation might leak information about the pair. In [dR91], de Rooij showed how to break the scheme. Schnorr proposed a fixed version in [Sch91], but this was also broken by de Rooij in [dR97].

In the precomputation phase of the fixed base exponentiation protocols, Tim needs to generate pairs (r, g^r) where r is random and g is the fixed base. This is the same as the precomputation that is needed for the *generation* of signatures in DSS [NIS94] and El Gamal based signature schemes, in particular Schnorr's scheme [Sch91]. We suggest that Tim also generate these pairs himself, by generating a random r, then computing g^r using a precomputation method [BGMW93,dR95,LL94]. The algorithms in [BGMW93,dR95,LL94] allow a time-space trade-off. The number of multiplications can be significantly reduced by using a modest amount of space and precomputation time.

4 Performance Analysis

4.1 Discrete Log-Based Signature Schemes

The problem of computing the discrete log (DL) in a cyclic group G of order n is a well known difficult problem in number theory. There are essentially two approaches to find discrete logarithms. The first approach extends G to a finite field $GF(q)$ where q is a prime power. The best known method to solve the discrete log in a field $GF(q)$ is a variant of the number field sieve [AD94,Cop84,Gor93,Kob94,Odl00,SWD96]. The heuristic expected asymptotic run time for this attack is $L[q, \frac{1}{3}, 1.923]$, where

$$L[q, v, u] = e^{(u+o(1))(\ln q)^v (\ln \ln q)^{1-v}}. \tag{1}$$

The second approach does not extend G to a finite field. In this case several methods are known, such as Pollard's birthday paradox based rho method [Pol78], to find the discrete log in G in $\mathcal{O}(\sqrt{n})$ operations.

We want to apply our exponentiation protocols to verifying signatures in signature schemes for which the security is based on the DL problem. Examples are DSS [NIS94], El Gamal's signature scheme [ElG85] and Schnorr's [Sch91] signature scheme. In these schemes, the exponentiation cannot be precomputed as it requires the message's signature. Not only can the exponentiation protocols be applicable to speeding up signature verification, but it can also be applied to speeding up exponentiation-based verification in general, such as the verification of Brands' digital credentials [Bra02].

The schemes we are considering use exponentiation modulo a large prime p. The security is based on the difficulty of finding the discrete log in the multiplicative group $\mathbb{Z}_p^* \cong GF(p)$, see (1). Notice that we do not need to keep the base or exponent secret while verifying signatures. For this reason we may apply our exponentiation protocols. The security parameter s of our protocol should match the security of the signature scheme itself. From (1) and Figure 3, this implies that we should choose

$$\log s = \log L[p, \frac{1}{3}, 1.923]$$
$$\approx 1.923(\log p)^{\frac{1}{3}}(\log\log p)^{\frac{2}{3}}.$$

This means that Tim needs $\mathcal{O}(\log s + \log w_s)$ online multiplications when he uses the untrusted Ursula to compute one exponentiation in $GF(p)$ without compromising the security of the signature scheme. Notice that the order of the multiplicative group is equal to $p-1 = w_s q_s$. To obtain the best performance we want $w_s \approx s$, that is the largest divisor ($= w_s$) of $p-1$ with prime factors at most $s \approx 2^{(\log p)^{\frac{1}{3}}}$ is at most s itself.

For simplicity we choose p to be a prime such that $(p-1)/2$ is prime as well ($(p-1)/2$ is called a Sophie-Germain prime). This means that $w_s = 1$. Table 1 shows the performance of variable base exponentiation without and with fixed exponent, where $k = \log p$ is the number of bits needed to represent p. For instance, in the case of variable base exponentiation without fixed exponent (Figure 3), with $w_s = 1$, Tim's online work mainly consists of computing $g^{m'}$, x^m, $z^{m'}$. The last two can be computed using simultaneous multiple exponentiation, which costs about 1.17 exponentiations [MvOV96, Algorithm 14.88], leading to an overall cost of $2.17 * \frac{3}{2} * \log s$ multiplications. Substituting for $\log s$, this leads to $6.3k^{\frac{1}{3}}(\log k)^{\frac{2}{3}}$ multiplications. If Tim computed g^a himself, using the square-and-multiply algorithm, it would cost him about $1.5k$ multiplications. Tim's speedup ($\frac{\texttt{work in square-and-multiply}}{\texttt{work in variable base exp protocol}}$) over the square-and-multiply algorithm is $0.24(\frac{k}{\log k})^{\frac{2}{3}}$. Ursula's work consists of computing two simultaneous multiple exponentiations, i.e. $2 * 1.17 * \frac{3}{2} = 3.51$ multiplications.

	without fixed exponent	with fixed exponent
Tim's online work	$6.3k^{\frac{1}{3}}(\log k)^{\frac{2}{3}}$	$5.8k^{\frac{1}{3}}(\log k)^{\frac{2}{3}}$
Tim's speedup over square-and-multiply	$0.24(\frac{k}{\log k})^{\frac{2}{3}}$	$0.26(\frac{k}{\log k})^{\frac{2}{3}}$
Tim's precomputation work	$\frac{3}{2}k$	$\frac{3}{2}k$
Tim's storage for precomputation (in bytes)	$\frac{k}{8}$	$\frac{k}{8}$
Ursula's work	$3.51k$	$1.75k$
Total bytes transmitted	$\frac{9}{8}k$	$\frac{5}{8}k$

Table 1. Performance of variable base exponentiation, without and with fixed exponent (the protocols in Figure 3 and Figure 4). $k = \log p$

4.2 Discussion on RSA

In the RSA signature scheme we compute modulo a composite integer $n = pq$ where p and q are large primes. The security of the scheme is based on keeping the factorization of n secret. If one wants to use our protocols of Subsection 3.3 then the security is also based

on the assumption that it is infeasible to compute the factorization of n given knowledge of w_s which divides $\phi(n)$ such that $\phi(n)/w_s$ does not contain any prime factors larger than s.

Computing the factorization of n without any side information is known to be a difficult problem, a method similar to the one which is used to solve the DL problem can be used. The heuristic expected asymptotic run time for this attack is $L[n, 1/3, 1.923]$. By the arguments of the previous subsection we need to choose a security parameter s such that $s \approx L[n, 1/3, 1.923]$. This leads to the same asymptotic results as presented in the previous subsection. It remains an open problem whether the additional assumption is valid.

We notice that the variable base exponentiation protocol with fixed exponent can be applicable to speeding up public-key message encryption if the base is blinded and the exponent is a fixed public encryption key which is known beforehand [Cha82]. Tim blinds the base which represents the message, using precomputed blinding factors, before using the protocols to get the assistance of Ursula. After Tim verifies the integrity of Ursula's operations, it can unblind the result to obtain the encrypted message. For the RSA cryptosystem, this application of our protocols may not be particularly useful because, in practice, the encryption exponent is small, enabling RSA encryption to be performed quickly.

One may be tempted to use a small private decryption key in RSA such that decryption is efficient. Then we may use our protocols to perform efficient online encryption with the in general large publicly known encryption key. However, this approach uses small decryption keys which is known to be broken [Wie90,BD00,DN00].

5 Related Work

As described in Section 1, there have been several proposals [MKI89,KS93,BQ95] for speeding up variable-base, fixed-exponent modular exponentiation using an untrusted resource, while trying to keep the exponent secret. Their main application was speeding up RSA [RSA78] signature generation on smart cards, in which case, the exponent is the trusted resource's secret key. Some of the proposals used precomputation and others did not. The proposals have been shown to be insecure. Successful attacks are able to, for example, return false values, or derive the secret key. Our variable base protocols are provably secure. However, we do not attempt to keep the exponent secret.

In [MBK00], Modadugu et al. demonstrate how to efficiently generate RSA keys on a low power handheld device with the aid of an untrusted server. Most of the key generation work is done by the server. However, the server must learn no information about the key it helps to generate. The paper presents a single server case, and a two-server case, where the two servers must not collaborate. In the single server case, partial-blinding is used to hide information on the keys being generated. They only use heuristics to argue the security of their schemes. We are solving different problems and our schemes are provably secure.

In [BQ94], Béguin and Quisquater present a protocol to accelerate variable base modular exponentiation where all of the parameters are public. They also present a protocol

to accelerate exponentiation modulo a prime number (not a composite integer), where the exponent can be kept secret. Their approach does not use precomputations. For moduli represented by 512 bits or less, their protocols achieve better gains compared to our protocols. Our protocols in Subsection 3.3 achieve better gains for large moduli. Our protocols use much less bandwidth for the whole range of possible moduli. We have also introduced, for the first time, protocols which compute exponentiations in arbitrary cyclic groups.

The protocol of Béguin and Quisquater uses techniques which can be applied in combination with our protocols. For example, our fixed base exponentiation protocol of Figure 1 can be extended by an additional phase in which we let Ursula compute x^m, x^{w_s}, and g^b by using the protocol of Béguin and Quisquater. Here, we put the commit and test paradigm in use. If Ursula is honest during the additional phase, Theorem 1 proves that the extended protocol is secure. If Ursula is dishonest during the additional phase then we rely on the security of Béguin and Quisquater's protocol. In essence Ursula committed to the algebraic equation $x^m g^r = y$ which Tim wants to test. After Ursula's commitment, uses Ursula's computing powers to perform the test.

Notice that in our variable base exponentiation protocol of Figure 3 we cannot compute $g^{m'}$ using the Béguin and Quisquater's protocol, where m' is made public. This is because m' needs to be kept secret up to the moment Tim performs his tests. It is possible to compute the other exponentiations using the Béguin and Quisquater's protocol.

The problem of checking the integrity of untrusted computation has also been addressed for other problems besides exponentiation. As described in Section 1, checkers in [BK89] can be used to check an untrusted resoure's computation on problems such as sorting and *gcd*. The problem is also important in the field of mobile security. One of the more challenging issues in this field is protecting mobile programs from untrusted hosts, as the host has complete control over the program. The model is a little different from our model in that the mobile program must have minimal communication with the trusted resource. The paper [ST98] presents a framework for how a mobile program might be able to securely sign the results of its computation on the untrusted resource: a valid signature is created only if the untrusted resource correctly executed the program. The framework suggests using function composition techniques; however, an actual provably secure scheme using their framework remains an open problem.

6 Conclusion

By using the commit and test paradigm we present protocols for speeding up fixed-base exponentiation and variable-base exponentiation using an untrusted computation resource. We are currently combining our protocols with XTR [LV00b,LV00a] and we are investigating into what extent Elliptic curve [D. 87] based signature schemes can be sped up by using our protocols.

References

[AD94] L. M. Adleman and J. DeMarrais. A subexponential algorithm for discrete logarithms over all finite fields. In *Advances in Cryptology - Crypto '93 Proceedings*, volume 773 of *LNCS*,

pages 147–158. Springer-Verlag, 1994.

[And92] R. J. Anderson. Attack on server-assisted authentication protocols. In *Electronic Letters*, 1992.

[BD00] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d les than $n^{0.292}$. In *IEEE Transactions on Information Theory*, volume 46 (4), pages 1339–1349, 2000.

[BDF+01] T. Berson, D. Dean, M. Franklin, D. Smetters, and M. Spreitzer. Cryptography as a network service. In *Network and Distributed System Security Symposium*, 2001.

[BGMW93] E. Brickell, D. M. Gordon, K. S. McCurley, and D. Wilson. Fast exponentiation with precomputation. In *Advances in Cryptology - Eurocrypt '92 Proceedings*, volume 658 of *LNCS*, pages 200–207. Springer-Verlag, 1993.

[BK89] M. Blum and S. Kannan. Designing programs that check their work. In *Proceedings of the* 21^{st} *Annual Symposium on Theory of Computing, ACM*, pages 86–97, 1989.

[BQ94] P. Béguin and J-J Quisquater. Secure acceleration of DSS signatures using insecure server. In *Advances in Cryptology - Asiacrypt '94 Proceedings*, volume 917 of *LNCS*. Springer-Verlag, 1994.

[BQ95] P. Béguin and J-J Quisquater. Fast server-aided RSA signatures secure against active attacks. In *Advances in Cryptology - Crypto '95 Proceedings*, volume 963 of *LNCS*, pages 57–69. Springer-Verlag, 1995.

[Bra02] Stefan Brands, 2002. http://www.credentica.com/technology/overview.pdf.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology - Crypto '82 Proceedings*, pages 199–203. Plenum Press, 1982.

[Cop84] D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. In *IEEE Trans. Inform. Theory 30*, pages 587–594, 1984.

[D. 87] D. Husemöller. *Elliptic Curves*. Springer-Verlag, 1987.

[DN00] G. Durfee and P. Nguyen. Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt '99. In *Advances in Cryptology - Asiacrypt 2000 Proceedings*, volume 1976 of *LNCS*, pages 14–29. Springer-Verlag, 2000.

[dR91] P. de Rooij. On the security of the Schnorr scheme using preprocessing. In *Advances in Cryptology - Eurocrypt '91 Proceedings*, volume 547 of *LNCS*, pages 71–80. Springer-Verlag, 1991.

[dR95] P. de Rooij. Efficient exponentiation using precomputation and vector addition chains. In *Advances in Cryptology - Eurocrypt '94 Proceedings*, volume 950 of *LNCS*, pages 389–399. Springer-Verlag, 1995.

[dR97] P. de Rooij. On Schnorr's preprocessing for digital signature schemes. *Journal of Cryptology*, 10(1):1–16, 1997.

[ElG85] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology - Crypto '84 Proceedings*, LNCS, pages 10–18. Springer-Verlag, 1985.

[Gor93] D. Gordon. Discrete logarithms in GF(p) using the number field sieve. In *SIAM J. Discrete Math. 6*, pages 312–323, 1993.

[Kob94] Neal Koblitz. *A Course in Number Theory and Cryptography, Second Edition*. Springer, 1994.

[KS93] S. Kawamura and A. Shimbo. Fast server-aided secret computation protocols for modular exponentiation. In *IEEE Journal on selected areas of communications*, volume 11, 1993.

[LL94] C. H. Lim and P. J. Lee. More flexible exponentiation with precomputation. In *Advances in Cryptology - Crypto '94 Proceedings*, volume 839 of *LNCS*, pages 95–107. Springer-Verlag, 1994.

[LV00a] Arjen K. Lenstra and Eric R. Verheul. An Overview of the XTR Public Key System. In *Proceedings of the Warsaw Conference on Public-Key cryptography and computational number theory*, 2000.

[LV00b] Arjen K. Lenstra and Eric R. Verheul. The XTR public key system. In *Advances in Cryptology - Crypto 2000 Proceedings*, volume 1880 of *LNCS*, pages 1–19. Springer-Verlag, 2000.

[MBK00] N. Modadugu, D. Boneh, and M. Kim. Generating RSA keys on a handheld using an untrusted server. In *Cryptographer's Track RSA Conference*, 2000.

[MKI89] T. Matsumoto, K. Kato, and H. Imai. Speeding up secret computation with insecure auxiliary devices. In *Advances in Cryptology - Crypto '88 Proceedings*, volume 403 of *LNCS*, pages 497–506. Springer-Verlag, 1989.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[NIS94] NIST. FIPS PUB 186: Digital Signature Standard, May 1994.

[NS98] P. Nguyen and J. Stern. The Béguin-Quisquater Server-Aided RSA Protocol from Crypto '95 is not Secure. In *Advances in Cryptology - Asiacrypt '98 Proceedings*, volume 1514 of *LNCS*, pages 372–379. Springer-Verlag, 1998.

[Odl00] A. Odlyzko. Discrete logarithms: The past and the future. In *Designs, Codes and Cryptography, 19*, pages 129–145, 2000.

[Pol78] J. M. Pollard. Monte Carlo methods for index computation (mod p). In *Math. Comp. 32*, pages 918–924, 1978.

[PW93] B. Pfitzmann and M. Waidner. Attacks on protocols for server-aided RSA computation. In *Advances in Cryptology - Eurocrypt '92 Proceedings*, volume 658 of *LNCS*, pages 153–162. Springer-Verlag, 1993.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sch90] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - Crypto '89 Proceedings*, volume 435 of *LNCS*, pages 239–252. Springer-Verlag, 1990.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[ST98] T. Sander and C. Tschudin. Towards mobile cryptography. In *IEEE Symposium on Security and Privacy*, 1998.

[SWD96] O. Schirokauer, D. Weber, and Th. F. Denny. Discrete logarithms: the effectiveness of the index calculus method. In *Proceedings ANTS II*, volume 1122 of *LNCS*. Springer-Verlag, 1996.

[Wie90] Michael J. Wiener. Cryptanalysis of short RSA secret exponents (abstract). In *IEEE Transactions on Information Theory*, volume 36 (3), pages 553–558, 1990.

A Proof of Theorem 1

Proof. Let α be a generator of the cyclic group G. Then any element in G can be expressed as a power of α and since n is the order of G, $1 = \alpha^n = \alpha^{w_s q_s}$. Suppose that instead of x and y Ursula transmits some tampered values $\alpha^u x$ and $\alpha^v y$ such that

- Tim's check passes and Ursula's tampering goes undetected, and
- Tim does not output the correct value g^a.

Since Tim would output the correct value $g^a = (\alpha^u x)^{w_s} g^b$ for all $u = 0$ modulo q_s, u is nonzero modulo q_s. Hence, by the Chinese remainder theorem, there exists a prime $p|q_s$ such that u is nonzero modulo p. Tim's check passes if and only if $(\alpha^u)^m = \alpha^v$, that is $um = v$ modulo n, in particular, $um = v$ modulo p (since $p|q_s|n$). This proves that Ursula is able to compute u and v such that $u \neq 0$ modulo p and $um = v$ modulo p. This is only possible if Ursula can guess or compute m modulo prime p. From $p|q_s$ we infer that $p \geq s$, hence $0 \leq m \leq s \leq p$. Thus Ursula can guess or compute m. Ursula's knowledge about m is given by $em + r$. Ursula does not know any other value involving m or r. Since r is a random number modulo n, $em + r$ does not reveal any information about m. This means that Ursula has guessed m correctly. This only happens with probability $1/s$ because m is a random number in $\{0, \ldots, s-1\}$.

B Proof of Theorem 2

Proof. The last two communication steps in Figure 3 is the fixed base protocol of Figure 1. Therefore, by Theorem 1, the probability that tampering with x goes undetected after checking $x^m\gamma^r = y$ is equal to $1/s$.

Let us assume that Ursula does not tamper with $x = \gamma^e$ (and y). This leaves only the values z and c to tamper with. Therefore, suppose that instead of z and c Ursula transmits some tampered values $\alpha^u z$ and $\alpha^v c$ such that

- Tim's check passes and Ursula's tampering goes undetected, and
- Tim does not output the correct value g^a.

Since Tim would output the correct value $g^a = (\alpha^u z)^{w_s} g^b$ for all $u = 0$ modulo q_s, u is nonzero modulo q_s. Hence, by the Chinese remainder theorem, there exists a prime $p|q_s$ such that u is nonzero modulo p. Since $x = \gamma^e$ is assumed not to be tampered with, Tim's check $z^{m'}x = c$ passes if and only if $(\alpha^u)^{m'} = \alpha^v$, that is $um' = v$ modulo n, in particular, $um' = v$ modulo p (since $p|q_s|n$). This proves that Ursula is able to compute u and v such that $u \neq 0$ modulo p and $um' = v$ modulo p. This is only possible if Ursula can guess or compute m' modulo prime p. From $p|q_s$ we infer that $p \geq s$, hence $0 \leq m' \leq s \leq p$. Thus Ursula can guess or compute m'. *At the moment* that Ursula tampers with z and c, Ursula's knowledge about m' is given by $h = g^{m'}\gamma$, Ursula does not know any other value involving m' or γ. Since $\gamma \in G$ is a random group element, $g^{m'}\gamma$ does not reveal any information about m'. This means that Ursula has guessed m' correctly. This only happens with probability $1/s$ because m' is a random number in $\{0, \ldots, s-1\}$.

C Proof of Security of Modular Exponentiation Protocol in Section 3.3

Proof. The proof of the security of the protocol is more complex (a ring may have zero divisors and may not be generated by a single element α). To prove the security we suppose that instead of x and y Ursula transmits some tampered values ux modulo n and vy modulo n such that

- Tim's check passes and Ursula's tampering goes undetected, and
- Tim does not output the correct value g^a modulo n.

Since Tim would output the correct value $g^a = (ux)^{w_s} g^b$ modulo n if $u^{w_s} = 1$ modulo n, we conclude that $u^{w_s} \neq 1$ modulo n. By the Chinese remainder theorem there exists a prime p_j such that $u^{w_s} \neq 1$ modulo $p_j^{e_j}$.

Lemma 1. *If $u^{w_s} \neq 1$ modulo $p_j^{e_j}$ then*

- *$p_j|u$ (that is, u is not invertible modulo $p_j^{e_j}$) or*
- *$u = \alpha_j^{u_j}$ modulo $p_j^{e_j}$, where α_j is a generator of the group of invertible integers modulo $p_j^{e_j}$ and $u_j w_s \neq 0$ modulo $\phi(p_j^{e_j})$ (hence, u_j is nonzero modulo $\phi(p_j^{e_j})/gcd(w_s, \phi(p_j^{e_j}))$).*

Proof. Notice that if u and p_j are relatively prime then u is an invertible integer modulo $p_j^{e_j}$. Since the invertible integers modulo $p_j^{e_j}$ form a group of order $\phi(p_j^{e_j})$, there exists a generator α_j which generates this group. This proves $u = \alpha_j^{u_j}$ modulo $p_j^{e_j}$ for some u_j. Since $u^{w_s} \neq 1$ modulo $p_j^{e_j}$, $u_j w_s \neq 0$ modulo $\phi(p_j^{e_j})$.

We first discuss the second case where u_j is nonzero modulo

$$\phi(p_j^{e_j})/gcd(w_s, \phi(p_j^{e_j})) \mid q_s.$$

By the Chinese remainder theorem, there exists a prime p dividing this number such that u_j is nonzero modulo p. Tim's check passes if and only if $u^m = v$ modulo n, in particular, $u^m = v$ modulo $p_j^{e_j}$. Since $u = \alpha_j^{u_j}$ is invertible modulo $p_j^{e_j}$, also $u^m = v$ is invertible modulo $p_j^{e_j}$. Therefore there exists an exponent v_j such that $v = \alpha_j^{v_j}$. From $u^m = v$ modulo $p_j^{e_j}$ we infer that $u_j m = v_j$ modulo $\phi(p_j^{e_j})$, in particular, $u_j m = v_j$ modulo p.

Now we continue with the proof of Theorem 1. We have shown that Ursula is able to guess or compute u_j and v_j such that $u_j \neq 0$ modulo p and $u_j m = v_j$ modulo p. This is only possible if Ursula can guess or compute m modulo prime p. From $p|q_s$ we infer that $p \geq s$, hence $0 \leq m \leq s \leq p$. Thus Ursula can guess or compute m. Ursula's knowledge about m is given by $em + r$. Ursula does not know any other value involving m or r. Since r is a random number modulo n, $em + r$ does not reveal any information about m. This means that Ursula has guessed m correctly. This only happens with probability $1/s$ because m is a random number in $\{0, \ldots, s-1\}$.

Let us now discuss the first case of Lemma 1 where $p_j|u$. Let us assume that if Ursula is able to compute or guess a prime factor p_j of n in feasible time then also Tim is able to precompute or guess the same prime factor p_j. If a prime factor p_j of n can be computed then a straighforward calculation reveals e_j. Therefore we assume that $n = wq$, where

- the factorization of w is possibly known to principals who have access to w, w_s and n, and
- it is infeasible to compute any prime factor of q given knowledge of w, w_s and n.

Clearly, tampering with u, where $p_j|u$ with $p_j|w|n$, can be detected by checking $gcd(x, w) = 1$ because

$$p_j|gcd(ux \mod n, w) \neq gcd(x, w) = gcd(g^e \mod n, w) = 1.$$

This observation leads to the protocol of Figure 5. If Tim wants to compute g^a modulo n for a general base g but with the restriction that n has only prime factors with single multiplicity (that is, each $e_j = 1$), then he may proceed as follows. He writes

$$g_1 = gcd(g, w) \text{ and } g_2 = g/g_1.$$

Since $gcd(g_2, w) = 1$, Tim can use the protocol in Figure 5 to compute $x_2 = g_2^a$ modulo n. If $a = 0$ then $g_1^a = 1$ and $g^a = 1$. If $a \neq 0$ then Tim uses the protocol in Figure 5 with n and w replaced by $n/gcd(g, w)$ and $w/gcd(g, w)$ to compute $x_1 = g_1^a$ modulo $n/gcd(g, w)$. This is possible since w consists of prime factors with single multiplicity such

that $gcd(g_1, w/gcd(g, w)) = 1$. Using the Euclidean algorithm Tim computes f such that $fg_1 = 1$ modulo $n/gcd(g, w)$. Now notice that

$$g_1^a = fg_1x_1 \mod n$$

(since $n/gcd(g, w)$ and $gcd(g, w)$ are relatively prime and the equation holds modulo $n/gcd(g, w)$ and modulo $g_1 = gcd(g, w)$, the equation also holds modulo n by the Chinese remainder theorem). This gives Tim the opportunity to compute

$$g^a = fg_1x_1x_2 \mod n$$

efficiently.