Breaking and Repairing Optimistic Fair Exchange from PODC 2003

Yevgeniy Dodis* Leonid Reyzin†

September 2, 2003

Abstract

In PODC 2003, Park, Chong, Siegel and Ray [22] proposed an optimistic protocol for fair exchange, based on RSA signatures. We show that their protocol is *totally breakable* already in the registration phase: the honest-but-curious arbitrator can easily determine the signer's secret key.

On a positive note, the authors of [22] informally introduced a connection between fair exchange and "sequential two-party multisignature schemes" (which we call *two-signatures*), but used an insecure two-signature scheme in their actual construction. Nonetheless, we show that this connection *can* be properly formalized to imply *provably secure* fair exchange protocols. By utilizing the state-of-the-art non-interactive two-signature of Boldyreva [6], we obtain an efficient and provably secure (in the random oracle model) fair exchange protocol, which is based on GDH signatures [9].

Of independent interest, we introduce a unified model for non-interactive fair exchange protocols, which results in a new primitive we call *verifiably committed signatures.* Verifiably committed signatures generalize (non-interactive) verifiably encrypted signatures [8] and two-signatures, both of which are sufficient for fair exchange.

1 Optimistic Fair Exchange

The problem of *fair exchange* is one of the fundamental problems in secure electronic transactions and digital rights management. Intuitively, it allows two parties to exchange items in a fair way, so that either each party gets the other's item, or neither party does. In the digital world, a natural instance of this problem is roughly the following. Alice is willing to sign some statement (e.g., e-cash payment, certified mail receipt, etc.), but only if Bob fulfills some obligation (delivers some good, discloses some information, etc.). On the other hand, Bob is not willing to fulfill this obligation unless he is sure that he gets the signature from Alice. A modern way to overcome this circularity is to introduces a semi-trusted *arbitrator* Charlie to the model. Alice will first register her key with Charlie. This registration is performed only once, and, as a result, Charlie may possibly learn some part of Alice's secret. Upon the completion of the one-time registration process, Alice can perform many fair exchanges with different merchants. In any such exchange, Alice first issues some verifiable "partial signature" σ' to Bob. Bob verifies the validity of this partial signature and fulfills his obligation by sending Alice the required information I, after which Alice sends her "full signature" σ to complete the

*Department of Computer Science, New York University, 251 Mercer Street, New York, NY 10012, USA. Email: dodis@cs.nyu.edu. Supported by the National Science Foundation under CAREER Award No. CCR-0133806 and Trusted Computing Grant No. CCR-0311095, and by the New York University Research Challenge Fund 25-74100-N5237.

†Department of Computer Science, Boston University, 111 Cummington St, Boston, MA 02215, USA. Email: reyzin@cs.bu.edu. Supported by the National Science Foundation under Grant No. CCR-0311485.

transaction. Thus, if no problem occurs, Charlie does not participate in the protocol (such protocols are called *optimistic*). However, if Alice refuses to send her full signature σ at the end, Bob will send σ' to Charlie (and a proof of fulfilling his obligation, including the information I that should be sent to Alice), and Charlie will convert σ' into σ, sending σ to Bob and I to Alice. Informally, we wish to achieve the following security guarantees:

- Alice should not be able to produce a valid partial signature σ' which Charlie cannot convert into a full signature σ.
- Bob should not be able to produce a valid partial (full) signature σ' (σ) which he did not get from Alice (Alice/Charlie provided Bob possesses σ').
- Charlie should not be able to produce a valid full signature σ without seeing a valid partial signature σ' computed by Alice.

While the first two properties are clearly important to prevent parties from cheating, the last property is equally crucial: we do not want the arbitrator Charlie to make signatures without Alice's consent. Indeed, otherwise Charlie would have to be completely trusted. Moreover, if one is willing to have a completely trusted arbitrator, then the problem becomes technically trivial, and no elaborate protocols (such as the protocol in [22] that we break) are needed at all: Alice may use any signature scheme and simply give Charlie her entire secret key during registration.

1.1 Previous Work

The problem of fair exchange has a rich history due to its fundamental importance. In the following, we only briefly mention the body of research most relevant to our results, and refer the reader to [2, 22] for further references.

Asokan et al. [1, 2] were the first to formally study the problem of optimistic fair exchange. They present several provably secure, but highly interactive solutions, based on the concept of *verifiably encrypted signatures* (VE-signatures). In such schemes, Alice encrypts her signature under Charlie's encryption key, and proves to Bob that she indeed encrypted her valid signature. After receiving her item from Bob, she proceeds to open the encryption. This approach of [1, 2] was later generalized by [10], but all these schemes involve expensive and highly interactive zero-knowledge proofs in the exchange phase. Other less formal works on interactive VE-signatures include [4, 3] (e.g., the paper of [4] was broken by [3]).

The first and only *non-interactive* VE-signature scheme was recently constructed by Boneh et al. [8]. While very elegant and provably secure (in the random oracle model), the scheme requires special elliptic curve groups with a bilinear map and relies on a form of the computational Diffie-Hellman assumption for such groups.[1]

A different paradigm for building *non-interactive* fair exchange protocols was very recently proposed by Park et al. [22]. Essentially, Alice commits by sending her "partial signature" σ' to Bob, and Bob is guaranteed that Charlie can convert it into Alice's full signature using the piece of Alice's secret

[1] As presented in [8], the scheme appears to also require a new and seemingly strong "aggregate extraction" security assumption; however, [13] shows that "aggregate extraction" assumption is equivalent to computational Diffie-Hellman.

that Charlie learned during Alice's registration.[2] The authors of [22] suggest that this approach may result in the design of simpler fair exchange protocols, and under more established security assumptions. To illustrate this point, they introduce a very efficient fair exchange protocol based on regular RSA signatures, and also informally sketch a deeper connection between their framework and "sequential two-party multisignature schemes" (which we call *two-signatures*). However, the authors of [22] provided no formal definitions for their framework, nor any proofs or even security arguments that their proposed protocol is secure.

1.2 Our Results

Unfortunately, we show that the fair exchange protocol presented by [22] at PODC 2003 (based on RSA signatures) is completely insecure. Specifically, we show that an honest-but-curious arbitrator Charlie can easily determine Alice's entire secret key after the end of Alice's registration. In other words, even though it might not seem at first that Alice leaks her entire key during registration, she effectively does so, thus trivializing the proposed scheme.

On a positive note, we show that the informal connection between non-interactive fair exchange and secure two-signature schemes *can* be formalized and result in provably secure fair exchange protocols, as long as one uses secure two-signature schemes (unlike the RSA-based scheme used by [22], which we show is completely insecure). In particular, by utilizing the state-of-the-art non-interactive two-signature of Boldyreva [6], we obtain a very efficient and provably secure (in the random oracle model) non-interactive fair exchange protocol, which is based on GDH signatures [9]. As compared to the non-interactive VE-signature of [8], the resulting fair exchange is equally efficient, but is based on a weaker "Gap Diffie-Hellman (GDH)" assumption and does not require a bilinear map.

We also stress that we provide formal definitions and security proofs for all our constructs. In particular, and of independent interest, we introduce a simple unified model for non-interactive fair exchange protocols: we model non-interactive fair exchange by a new primitive we call *verifiably committed signatures.* Verifiably committed signatures generalize (non-interactive) verifiably encrypted signatures [8] and two-signatures, both of which are sufficient for fair exchange.

2 Formal Model For Non-Interactive Fair Exchange

We introduce the concept of *verifiably committed signatures*,[3] which directly model non-interactive fair exchange.

Definition 1 *A* verifiably committed signature *(equivalently,* non-interactive fair exchange*) involves the signer Alice, the verifier Bob and the arbitrator Charlie, and is given by the following efficient*

[2] At first glance, it may seem that this approach has a major drawback in that Charlie would have to store a secret (learned during the user's registration) for *each* user Alice in the system (unlike the VE-signature approach, where Charlie has to store a single encryption key, independent of each user's information.) However, as mentioned in [22, Remark 5], Charlie can encrypt the secret share learned from each user via any secure symmetric encryption, thus replacing secret storage with public storage. Furthermore, Charlie can "delegate" the responsibility for storing this encrypted secret information back to the user, by including it as part of Alice's certificate, which Alice must anyway receive during registration and send with her partial signatures.

[3] Our notion is very different from "signatures on committed values" (see [11]). There, one tries to hide the message signed, but interactively prove that the message satisfies some property. It is also different from designated-confirmer signatures [12], because the trusted party completes the signature to be verifiable by anyone, rather than confirming it in a zero-knowledge protocol.

procedures:

- Setup. *This is an interactive protocol between Alice and Charlie, by the end of which either (1) one of the parties aborts, or (2) Alice learns her secret signing key* SK*, Charlie learns his secret arbitration key* ASK*, and both parties agree on Alice's public verification key* PK*, and partial verification key* APK.

- Sig *and* Ver. *These are conventional signing and verification algorithms of an ordinary signature scheme.* $\mathsf{Sig}(m, \mathsf{SK})$ *— run by Alice — outputs a signature* σ *on* m*, while* $\mathsf{Ver}(m, \sigma, \mathsf{PK})$ *— run by Bob (or any verifier) — outputs* 1 *(accept) or* 0 *(reject).*

- PSig *and* PVer. *These are partial signing and verification algorithms, which are just like ordinary signing and verification algorithms, except they can depend on the public arbitration key* APK. $\mathsf{PSig}(m, \mathsf{SK}, \mathsf{APK})$ *— run by Alice — outputs a partial signature* σ'*, while* $\mathsf{PVer}(m, \sigma', \mathsf{PK}, \mathsf{APK})$ *— run by Bob (or any verifier) — outputs* 1 *(accept) or* 0 *(reject).*

- Res. *This a resolution algorithm run by Charlie in case Alice refuses to open her signature* σ *to Bob, who in turn possesses a valid partial signature* σ' *on* m *(and a proof that he fulfilled his obligation to Alice). In this case,* $\mathsf{Res}(m, \sigma', \mathsf{ASK}, \mathsf{PK})$ *should output a legal signature* σ *of* m.

Correctness states that $\mathsf{Ver}(m, \mathsf{Sig}(m, \mathsf{SK}), \mathsf{PK}) = 1$*,* $\mathsf{PVer}(m, \mathsf{PSig}(m, \mathsf{SK}, \mathsf{APK}), \mathsf{PK}, \mathsf{APK}) = 1$ *and* $\mathsf{Ver}(m, \mathsf{Res}(\mathsf{PSig}(m, \mathsf{SK}, \mathsf{APK}), \mathsf{ASK}, \mathsf{PK}), \mathsf{PK}) = 1$. *Moreover, we strengthen the last equation and require that the distribution of any "resolved signature"* $\mathsf{Res}(\mathsf{PSig}(m, \mathsf{SK}, \mathsf{APK}), \mathsf{ASK}, \mathsf{PK})$ *is identical to (or at least computationally indistinguishable from) the "actual signature"* $\mathsf{Sig}(m, \mathsf{SK})$.

A few remarks are in order. The key PK is Alice's long-term public key; therefore, in a typical application, it will be certified by some certificate authority as belonging to Alice and Alice will be held responsible for signatures under PK. On the other hand, APK should carry no legal meaning outside of the arbitration process. In particular, σ' should not be viewed as Alice's signature, even though it can be publicly verified using PK and APK, because it does not mean that Bob has fulfilled his obligations to Alice.

The definition above assumes that Bob, when verifying the partial signatures, has the authentic copies of PK and APK produced during the Setup procedure. He cannot, in general, simply trust Alice to give him correct PK and APK, because a cheating Alice may modify the keys produced during setup and attempt to use public keys for which the Res procedure does not work. Bob needs to make sure that Charlie agrees to these keys. This can be done in a variety of ways, the most convenient of which depends on the application and the scheme. For example, Charlie could have his own public key with which he can certify PK and APK at the end of Setup. Alternatively, Charlie can provide APK to Bob directly (this is most convenient when there is a single APK for all users, as in the case of verifiably encrypted signatures below).

In a meaningful application Charlie should run Res to produce a full signature σ from σ' only upon ensuring that Bob's obligation to Alice has been fulfilled. Our definition does not deal with the application-specific question of how Bob can prove to Charlie that he fulfilled his obligations to Alice. For example, such proof could be Bob's digital signature on some contract, in which case Charlie will also forward this signature to Alice before giving Alice's signature to Bob. We also observe that our framework does not address the subtle issue of timely termination addressed by [1, 2]. We remark, however, that the technique of [1, 2] can be easily added to our solution to resolve this problem.

Finally, the fact the resolved signatures look identical to the real signatures implies that outside parties are not able to tell whether or not the resolution process took place, which is typically very desirable. In particular, this rules out some trivial solutions where the verifier can either accept a real signature by Alice, or a separate signature by Charlie (in conjunction with Alice's verifiably committed signature).

We now comment on the known paradigms for implementing verifiably committed signatures.

2.1 Verifiably Encrypted Signatures

In a setup that requires the least interaction, Charlie generates $(\mathsf{ASK}, \mathsf{APK})$ by himself, while Alice generates $(\mathsf{SK}, \mathsf{PK})$ by herself (possibly depending on APK). This way Charlie can support many users with a single arbitration key ASK, and Alice does not need to contact Charlie at all when she produces her keys (she even does not need her keys to be certified by Charlie).

A natural approach to implementing this is to have $(\mathsf{APK}, \mathsf{ASK})$ be Charlie's encryption/decryption keys. Then Alice generates the partial signature σ' by encrypting her actual signature σ using APK. If needed, Charlie resolves this by simply decrypting the "verifiably encrypted signature" σ'. We will refer to such special case of verifiably committed signature as (non-interactive) *verifiably encrypted signature* (or VE-signature).

The challenge of this approach is to make σ' *non-interactively* verifiable. Indeed, until very recently [8], all VE-signatures were highly interactive (between Alice and Bob). The only non-interactive VE-signature of [8] explicitly requires a special group with a bilinear map, and is based (in the random oracle model) on a variant of the Diffie-Hellman assumption in such a group [13].

2.2 Two-Signatures

An alternative paradigm was explicitly suggested by Park et al. [22]. In this case, *all four keys* $\mathsf{SK}, \mathsf{PK}, \mathsf{ASK}, \mathsf{APK}$ are generated by Alice. Then, Alice sends $\mathsf{PK}, \mathsf{ASK}, \mathsf{APK}$ to Charlie, who checks if the keys were "properly generated". Ideally, this check should be non-interactive (which will be the case in our later solution), but Park et al. also allow Alice to interactively prove the correctness of $\mathsf{PK}, \mathsf{ASK}, \mathsf{APK}$ (e.g., prove her knowledge of SK). Charlie will then certify Alice's key APK, and Bob will accept partial signatures from Alice only if they are made using certified APK.

Charlie can store ASK for each user Alice, but this obviously requires a lot of secret storage that the previous approach does not. However, this is not necessary, as shown in [22]. To avoid this large secret storage, Charlie can generate a secret key K for any semantically secure symmetric cryptosystem (E, D), and store $E_K(\mathsf{ASK})$ for each user, thus converting secret storage to public storage. Moreover, Charlie can then include $E_K(\mathsf{ASK})$ into the certificate he issues for APK, thus avoiding the need for public storage entirely (because the certificate for APK must be included in every partial signature σ, anyway).

As for building the actual verifiably committed signature scheme using this approach, Park et al. suggest to use "sequential two-party multisignatures" (from now on, referred to as *two-signatures*). In such signatures, two parties P_1 and P_2 have two pairs of keys (pk_1, sk_1) and (pk_2, sk_2). They can then jointly sign a message m by first having P_1 sign m using sk_1 (producing σ_1), and then P_2 transform σ_1 into a "joint" signature σ which the verifier can check was signed by both P_1 and P_2. For that, the verifier uses the "combined" public key $pk = pk_1 \star pk_2$, where $\star$ is some public operation (concatenation always works, but one usually wants to have pk of the same length as pk_1 and pk_2). Moreover, it is often the case that the secret keys sk_1 and sk_2 can also be "combined" into a joint secret key $sk = sk_1 \otimes sk_2$ which can be used to generate σ "directly". In this case, one can use such a two-signature as a verifiably committed signature by setting $\mathsf{SK} = (sk, sk_1)$, $\mathsf{PK} = pk$, $\mathsf{ASK} = sk_2$ and $\mathsf{APK} = pk_1$ (notice, nobody needs to know pk_2), Sig to be the direct signing using sk, PSig to be the partial signing by P_1 using sk_1 (so that $\sigma' = \sigma_1$), and Res to be the signature completion performed by P_2 using sk_2.[4]

As compared to VE-signatures, the obvious disadvantage of this approach is the need for interaction between Alice and Charlie, where Charlie needs to certify Alice's APK. However, this interaction is performed only once for each user Alice. An advantage of the approach is that it can lead to simpler solutions under weaker assumptions.

For instance, a trivial (but secure!) example of this paradigm will be to have two arbitrary signature schemes with keys (pk_1, sk_1), (pk_2, sk_2), and let $\mathsf{PK} = (pk_1, pk_2)$, $\mathsf{SK} = (sk_1, sk_2)$ and $\sigma = (\sigma_1, \sigma_2)$. In other words, Alice's regular signature consists of two independent signatures σ_1 and σ_2, while Charlie knows how to produce σ_2 only. Then Alice commits to her signature by sending σ_1 to Bob, who will then ask Charlie to produce σ_2 if Alice refuses to do so later. Of course, such two-signatures / fair exchange protocols are uninteresting and inefficient. First, all the computation and key/signature lengths have to be doubled, and, more importantly, the final signature σ is not consistent with some existing "natural" signature scheme. In particular, for the uses outside of the fair exchange framework, Alice has to suffer with an inefficient and non-standard signature scheme. Thus, for the purposes of efficiency and compatibility (but not security!), the goal is to design two-signature schemes which are consistent with some natural, "atomic" signature schemes. Still, the trivial solution above *should* satisfy the security properties of verifiably committed signatures, and thus serve as an inspiration for our formal definition of verifiably committed signatures below.

2.3 Security of Verifiably Committed Signatures

The security of verifiably committed signatures consists of ensuring three aspects: security against signer Alice, security against verifier Bob, and security against arbitrator Charlie. In the following, we denote by P an oracle simulating the partial signing procedure PSig, and by R — the oracle simulating the resolution procedure Res. Also, k denotes the security parameter, and PPT stands for "probabilistic polynomial time" (in the security parameter).

Security against Alice. We require that any PPT adversary A succeeds with at most negligible probability in the following experiment.

$$\begin{array}{rcl}
\mathsf{Setup}^*(1^k) & \rightarrow & (\mathsf{SK}^*, \mathsf{PK}, \mathsf{ASK}, \mathsf{APK}) \\
(m, \sigma') & \leftarrow & A^R(\mathsf{SK}^*, \mathsf{PK}, \mathsf{APK}) \\
\sigma & \leftarrow & \mathsf{Res}(m, \sigma', \mathsf{ASK}, \mathsf{PK}) \\
\text{success of } A & = & [\mathsf{PVer}(m, \sigma', \mathsf{PK}, \mathsf{APK}) \overset{?}{=} 1 \ \wedge \ \mathsf{Ver}(m, \sigma, \mathsf{PK}) \overset{?}{=} 0]
\end{array}$$

where Setup^* denotes the run of Setup with dishonest Alice (run by A) and SK^* is A's state after this run. In other words, Alice should not be able to produce partial signature σ' which looks good to Bob,

[4] Note that in this application of multisignatures, P_1 generates both keys, and hence there is no danger of "active insider attacks" (where some dishonest party generates its key maliciously so as to be able to forge signatures on behalf of the rest of the group). In particular, two-signatures needed here are weaker than the accountable-subgroup multisignatures of [21].

but which cannot be transformed into Alice's full signature by honest Charlie. As mentioned above, this assumes that Bob has authentic copies of PK and APK produced during setup.

Security against Bob. We require that any PPT adversary B succeeds with at most negligible probability in the following experiment.

$$\begin{array}{rcl} \mathsf{Setup}(1^k) & \rightarrow & (\mathsf{SK}, \mathsf{PK}, \mathsf{ASK}, \mathsf{APK}) \\ (m, \sigma) & \leftarrow & B^{P,R}(\mathsf{PK}, \mathsf{APK}) \\ \text{success of } B & = & [\mathsf{Ver}(m, \sigma, \mathsf{PK}) \overset{?}{=} 1 \ \wedge \ (m, \ldots) \notin \mathit{Query}(B, R)] \end{array}$$

where $\mathit{Query}(B, R)$ is the set of *valid* queries B asked to the resolution oracle R (i.e., (m, σ') such that $\mathsf{PVer}(m, \sigma') = 1$). In other words, Bob should not be able to complete any of the partial signatures σ' that he got from Alice into a complete signature σ, without explicitly asking Charlie to do that (in which case he must have been completed his obligation, since otherwise we assumed that Charlie would not cooperate).

Notice also that there is no need to provide B with an oracle access to the signing oracle Sig, since it could be simulated by P and R. Finally, we remark that we also want Bob to be unable to generate a valid σ' which was not produced by Alice (via a query to $P = \mathsf{PSig}$). However, this guarantee will always follow from an even stronger security against Charlie, which we define below. Indeed, we will ensure that even Charlie — who knows more than Bob (i.e., ASK) — cannot succeed in this attack.

Security against Charlie. We require that any PPT adversary C succeeds with at most negligible probability in the following experiment.

$$\begin{array}{rcl} \mathsf{Setup}^*(1^k) & \rightarrow & (\mathsf{SK}, \mathsf{PK}, \mathsf{ASK}^*, \mathsf{APK}) \\ (m, \sigma) & \leftarrow & C^{P}(\mathsf{ASK}^*, \mathsf{PK}, \mathsf{APK}) \\ \text{success of } C & = & [\mathsf{Ver}(m, \sigma, \mathsf{PK}) \overset{?}{=} 1 \ \wedge \ m \notin \mathit{Query}(C, P)] \end{array}$$

where Setup^* denotes the run of Setup with dishonest Charlie (run by C), ASK^* is C's state after this run, and where $\mathit{Query}(C, P)$ is the set of queries C asked to the partial signing oracle P. In other words, Charlie should not be able to produce a valid signature on m of Alice without explicitly asking Alice to produce a partial signature on m (which he can complete into a full signature by himself using ASK).

We remark that this property is crucial. Even though Charlie is semi-trusted, Alice does not want Charlie to produce valid signatures which she did not intend on producing (otherwise, fair exchange would become trivial: Alice could just give Charlie her secret key). As we will see in Section 3, the verifiably committed signature of Park et al. [22] fails to achieve this property: in fact, an honest-but-curious Charlie can completely determine Alice's entire secret key SK, without any queries to the partial signing oracle!

Finally, we remark that since Bob's information is subsumed by either Alice's or Charlie's information, there is no need to consider a coalition of Alice (Charlie) and Bob attacking Charlie (Alice). On the other hand, Bob is certainly not protected if Alice and Charlie collude, as Charlie can refuse to resolve the signature. Thus, our definition is the most general one can hope to achieve.

Novelty of Our Security Model. We believe that our precise and formal definition of verifiably committed signatures is of independent interest. While previous work (such as [1, 2]) gave elaborate formal definitions of interactive fair exchange, ours is the first clean and simple definition of non-interactive fair exchange. In particular, our definition is not just a "trivial extension" (to a more general Setup procedure) of the definition of non-interactive VE-signatures from Boneh et al. [8]. Indeed, the former paper gave a nice and formal definition of VE-signatures, but did not explicitly consider security against Alice and Charlie (instead, it considered only two forms of security against Bob). Even though we believe that the scheme in [8] satisfies our more general definition — and the proofs should easily follow from the ones given in [8] for their weaker definition — our *definition* is a noticeable strengthening over the one given in [8]. Additionally, our definition unifies the framework of VE-signatures and that of two-signatures informally presented by [22] (the latter work had no formal definitions at all).

3 Breaking the Fair Exchange Scheme from PODC 2003

As we mentioned, Park et al. [22] suggested a fair exchange scheme based on the two-signature paradigm described in Section 2. Specifically, they attempted to build a two-signature scheme based on regular RSA full domain hash signature scheme [5]. Recall, in this scheme one chooses the modulus $n = pq$ to be a product of two (safe) k-bit primes, chooses a random public key $e \in \mathbb{Z}^*_{\varphi(n)}$ (where $\varphi(n) = (p-1)(q-1)$), and sets the secret key $d \equiv e^{-1} \bmod \varphi(n)$. To sign a message m, one computes $\sigma \equiv H(m)^d \bmod n$, where H is a secure hash function (modeled as a random oracle). To verify, one checks that $\sigma^e \equiv H(m) \bmod n$.

Based on this RSA signature, Park et al. attempted to build the following two-signature scheme. Randomly split $d \equiv d_1 + d_2 \bmod \varphi(n)$ (where $d_1 \in \mathbb{Z}^*_{\varphi(n)}$), let $e_1 \equiv d_1^{-1} \bmod \varphi(n)$ and, following the notation of Section 2, set $pk = e$ (we omit n, which is implicitly given), $pk_1 = e_1$, $sk_1 = d_1$, $sk_2 = d_2$. Notice, we indeed have

$$\sigma \equiv H(m)^d \equiv H(m)^{d_1} \cdot H(m)^{d_2} \equiv \sigma_1 \cdot \sigma_2 \bmod n$$

so that Alice can commit to σ by sending σ_1, and Charlie can complete it into the full signature σ — if necessary — by knowing d_2 and computing $\sigma_2 \equiv H(m)^{d_2} \bmod n$.

Notice, however, that in this scheme Charlie knows n, e, e_1 and d_2. Had Charlie also known $pk_2 = e_2 \equiv d_2^{-1} \bmod \varphi(n)$, then the scheme would be obviously insecure since the integer $(e_2 d_2 - 1)$ would be a non-zero multiple of $\varphi(n)$, and it is well known that knowing such multiple of $\varphi(n)$ is equivalent to factoring n (e.g., page 94 of [19]). "Luckily", the authors of [22] observed that there is no need to give e_2 to Charlie, so the system "is still secure". Unfortunately, we show that this claim is false. Even without knowing e_2, an honest-but-curious Charlie can still determine a non-zero multiple of $\varphi(n)$, and thus factor n. We now summarize our break into the following abstract problem.

PROBLEM: Pick two random k-bit primes p, q and set $n = pq$. Pick two random RSA key pairs (d, e) and (d_1, e_1): namely, choose random $e, e_1 \in \mathbb{Z}^*_{\varphi(n)}$, and set $d \equiv e^{-1} \bmod \varphi(n)$, $d_1 \equiv e_1^{-1} \bmod \varphi(n)$. Let $d_2 \equiv d - d_1 \bmod \varphi(n)$. The problem is to factor n given n, e, e_1, d_2.

Theorem 1 *The problem above can be solved in probabilistic polynomial time. Thus, an honest-but-curious arbitrator Charlie can easily determine Alice's secret key at the end of the setup procedure.*

Proof: Since $ed \equiv 1 \bmod \varphi(n)$ and $d \equiv d_1 + d_2 \bmod \varphi(n)$, we have $ed_1 \equiv (1 - ed_2) \bmod \varphi(n)$. Multiplying by e_1 and using $e_1 d_1 \equiv 1 \bmod \varphi(n)$, we have

$$e \equiv (1 - ed_2)e_1 \bmod \varphi(n) \tag{1}$$

Notice, all the quantities e, e_1 and d_2 are given to us in the problem statement. Moreover, they are given to us as positive integers. Thus, $e > 0$ and $(1 - ed_2)e_1 \leq 0$. This means that Equation (1) above *cannot hold over the integers*, which in turn means that the integer

$$I \stackrel{\text{def}}{=} e - (1 - ed_2)e_1$$

is a *non-zero* multiple of $\varphi(n)$. However, we already mentioned that a knowledge of a non-zero multiple of $\varphi(n)$ is sufficient to factor n, which completes the proof. □

MULTIPLICATIVE SHARING. Interestingly, the authors of [22] noticed that their scheme would be insecure if d is split *multiplicatively*, even though their argument was somewhat incomplete. For thoroughness, we give the full argument, since it is very short anyway. Notice, in this case $d_2 \equiv dd_1^{-1} \bmod \varphi(n)$, which is equivalent to $d_2 e \equiv e_1 \bmod \varphi(n)$. Thus, the integer $J \stackrel{\text{def}}{=} (d_2 e - e_1)$ is a multiple of $\varphi(n)$. However, to factor n we still have to show that $J \neq 0$, which the authors of [22] did not do. But this argument is simple as well. Indeed, since e and e_1 were chosen randomly, with all but negligible probability we have $d_2 > 2^k$, $e > 2^k$, while clearly $e_1 < n < 2^{2k}$. But this means that with all but negligible probability $J > 0$, so J is a non-zero multiple of $\varphi(n)$ indeed.

4 Secure Fair Exchange based on GDH Signatures

The break on the scheme from PODC 2003 was due to the fact that the authors utilized an insecure two-signature scheme in their construction. In this section we show that one can build secure verifiably committed signatures provided one uses secure two-signatures. One way to approach this claim would be to give a formal definition of secure two-signatures (i.e., "sequential two-party multisignatures"). However, the resulting definition would be essentially the same as the definition of verifiably committed signatures we are trying to satisfy (except it will use a particular form of the Setup procedure), and the whole implication will anyway essentially be a tautology. Moreover, there currently anyway exists only one fully non-interactive multisignature scheme of [6], where the underlying signature is consistent with an existing "atomic" signature scheme of [9]. Thus, there does not seem to be a justifiable reason to give a complicated ad hoc definition of a new primitive, which is not much easier to satisfy than that of verifiably committed signatures, and of which we anyway currently have only one example.

Therefore, we choose to give a more meaningful direct adaptation of the multisignature scheme of Boldyreva [6] into a verifiably committed signature scheme, and then prove that the resulting scheme satisfies our formal definitions from Section 2.3. We remark that our proof is quite simple, but does not immediately follow from the one given by [6], since our model and security notion are new and different.[5] First we introduce "gap Diffie-Hellman (GDH) Groups" [18, 17] and the corresponding GDH signature scheme [9].

GDH GROUPS. Assume G is a multiplicative group of prime order p. Consider the following two problems in G.

Computational Diffie-Hellman (CDH) Problem: given three elements $g, h, u \in G$, compute $v = u^{\log_g h}$.

[5] As we stated earlier, one probably could make our result "generically follow" from the multisignature security of [6], but it is much easier to prove it directly.

Decisional Diffie-Hellman (DDH) Problem: given four elements $g, h, u, v \in G$, determine whether or not they satisfy the relation $\log_g h = \log_u v$ (in case they do, the tuple (g, h, u, v) is called the *DDH-tuple*).

We can now define the GDH groups. Basically, in these groups the DDH problem is easy, but the CDH problem is assumed to be hard. Below, we assume that there exists a family of the corresponding groups G parameterized by some security parameter k, and efficiency is measured in terms of the binary length of the group order p (which is polynomial in k).

Definition 2 *A prime order group G (with efficient group operation and its inverse) is called a GDH group if there exists an efficient polynomial time algorithm $\mathcal{V}_{\mathsf{DDH}}$ which solves the DDH problem, but no* PPT *algorithm can solve the CDH problem with non-negligible probability, when the inputs g, h, u are chosen at random.*

We remark that GDH groups have found many applications recently (e.g., [7, 9, 16, 20, 8, 6, 15]).

GDH Signatures. The GDH signature scheme in a GDH group G is defined as follows. The key generation algorithm picks a GDH group G of order p, and random $g \in G$, $x \in \mathbb{Z}_p$. It computes $h = g^x$, and sets the public key to be (g, h) (G and p are assumed to be public parameters too), and the secret key to be x. To sign a message m, one computes $\sigma = H(m)^x$, where H is a random oracle. To verify σ, one outputs $\mathcal{V}_{\mathsf{DDH}}(g, h, H(m), \sigma)$, i.e., tests if $(g, h, H(m), \sigma)$ form a DDH-tuple. This scheme can also be viewed as a slight generalization of the full domain hash paradigm [5].

Theorem 2 ([9]) *If G is a GDH group, then the GDH signature above is existentially unforgeable under adaptive chosen message attack in the random oracle model.*

As observed by [9], the GDH signatures not only give yet another simple and efficient signature scheme under a new assumption, but also have the advantage of being very short in the currently proposed GDH groups.

4.1 Verifiably Committed Signature Based on GDH Signatures

We now extend the above signature into a verifiably committed signature, following [6].

- Setup. Alice chooses random $g \in G$, $x, x_1 \in \mathbb{Z}_p$, computes $x_2 = x - x_1 \bmod p$, $h = g^x$, $h_1 = g^{x_1}$, and sets $\mathsf{PK} = (g, h)$, $\mathsf{SK} = (x, x_1)$, $\mathsf{APK} = h_1$, $\mathsf{ASK} = x_2$. She then sends $\mathsf{PK}, \mathsf{APK}, \mathsf{ASK}$ to Charlie, who checks that $h = h_1 g^{x_2}$ (and rejects if this is not the case).
- Sig and Ver are identical to the GDH signature: $\mathsf{Sig}(m) = H(m)^x$, $\mathsf{Ver}(m, \sigma) = \mathcal{V}_{\mathsf{DDH}}(g, h, H(m), \sigma)$.
- PSig and PVer are also identical to the GDH signature, but with public key h_1: $\mathsf{PSig}(m) = H(m)^{x_1}$, $\mathsf{PVer}(m, \sigma') = \mathcal{V}_{\mathsf{DDH}}(g, h_1, H(m), \sigma')$.
- $\mathsf{Res}(m, \sigma')$ first checks that $\mathsf{PVer}(m, \sigma') = 1$, and then outputs $\sigma = \sigma' H(m)^{x_2}$.

The correctness property of the above verifiably committed signature is obvious. We now analyze its security.

Theorem 3 *The GDH verifiably committed signature above is as secure as the regular GDH signature. In particular, it is secure in GDH groups in the random oracle model.*

Proof: We prove the security against Alice, Bob and Charlie.

Security against Alice follows unconditionally. Indeed, if Charlie accepted the values (g, h, h_1, x_2) in the registration, it means that $x \overset{\text{def}}{=} \log_g h$ and $x_1 \overset{\text{def}}{=} \log_g h_1$ satisfy $x_1 + x_2 = x \bmod p$. Also, any valid partial signature σ' satisfies $x_1 = \log_g h_1 = \log_{H(m)} \sigma'$, and therefore the resolved full signature $\sigma = \sigma' H(m)^{x_2}$ satisfies $\log_{H(m)} \sigma = x_1 + x_2 = x = \log_g h$, and thus must pass the usual verification algorithm.

To show security against Bob, we convert any attacker B that attacks our verifiably committed signature into a forger F for the regular GDH signature. Recall, F gets (g, h) as an input, and has oracle access to the signing oracle Sig. On the other hand, B expects (g, h, h_1) and oracle access to both PSig and Res, and wins if it forges a signature σ of some message m without asking Res a valid query (m, σ'). Since there is only one valid σ' for a given m and B can test the validity himself, we can assume that B simply did not ask Res any queries involving the forged message m.

So here is how F simulates the run of B. It picks a random $x_1 \in \mathbb{Z}_p$, sets $h_1 = g^{x_1}$ and gives (g, h, h_1) to B. F can respond to PSig queries of B by himself, since he knows x_1. To simulate a valid resolution query (m_i, σ'_i) to Res, F simply asks its own signing oracle on message m, and returns the answer to B. When B outputs the forgery (m, σ), F also outputs the same forgery. We see that the simulation is perfect, and F succeeds in producing a new forgery if and only if B succeeds.

Finally, we show security against Charlie. Again, we convert any verifiably committed signature attacker C into a forger F for the regular GDH signature. As before, F gets (g, h) as an input, and has oracle access to the signing oracle Sig. On the other hand, C expects (g, h, h_1, x_2) and oracle access to PSig, and wins if it forges a signature σ of some message m without asking $\mathsf{PSig}(m)$.

So here is how F simulates the run of C. It picks a random $x_2 \in \mathbb{Z}_p$, sets $h_1 = hg^{-x_2}$ and gives (g, h, h_1, x_2) to C. F can respond to PSig queries m_i of C by first getting a signature $\sigma_i = H(m_i)^x$ from its own signing oracle, and then returning $\sigma'_i = \sigma_i H(m_i)^{-x_2}$. When C outputs the forgery (m, σ), F also outputs the same forgery. We see that the simulation is perfect, and F succeeds in producing a new forgery if and only if C succeeds. □

Remark 1 *It is instructive to see where the above proof of security against Charlie fails for seemingly very similar RSA signatures. The step that fails involves computing the public arbitration key* $e_1 \equiv e(d_2)^{-1} \bmod \varphi(n)$ *from the global public key* e *and a random* d_2 *which the simulator* F *chooses. Indeed, a natural way for doing so involves computing the inverse of* d_2 *modulo* $\varphi(n)$*, which is as hard as factoring* n*. In fact, our break shows that there is* no way *to complete the reduction, unless the standard RSA assumption is false.*

Acknowledgments

We thank Alexandra Boldyreva, Jung Min Park, and Victor Shoup for useful discussions concerning fair exchange and comments on this paper.

References

[1] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In K. Nyberg, editor, *Advances in Cryptology—EUROCRYPT 98*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer-Verlag, May 31–June 4 1998.

[2] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communication*, 18(4):593–610, 2000.

[3] G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In G. Tsudik, editor, *Sixth ACM Conference on Computer and Communication Security*, pages 138–146. ACM, Nov. 1999.

[4] F. Bao, R. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line TTP. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 77–85, 1998.

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communication Security*, pages 62–73, November 1993. Revised version appears in `http://www-cse.ucsd.edu/users/mihir/papers/crypto-papers.html`.

[6] A. Boldyreva. Efficient threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. In Desmedt [14].

[7] D. Boneh and M. Franklin. Identity based encryption from the weil pairing. In J. Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 19–23 Aug. 2001.

[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology—EUROCRYPT 2003*, Lecture Notes in Computer Science, pages 416–432. Springer-Verlag, 4 May–8 May 2003.

[9] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In C. Boyd, editor, *Advances in Cryptology—ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, 9–13 Dec. 2001. Springer-Verlag.

[10] J. Camenisch and I. B. Damgård. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In T. Okamoto, editor, *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345, Kyoto, Japan, 3–7 Dec. 2000. Springer-Verlag.

[11] J. Camenisch and A. Lysyanskaya. Signature schemes with efficient protocols. In *Conference on Security in Communication Networks (SCN)*, 2002.

[12] D. Chaum. Designated confirmer signatures. In A. De Santis, editor, *Advances in Cryptology—EUROCRYPT 94*, volume 950 of *Lecture Notes in Computer Science*, pages 86–91. Springer-Verlag, 1995, 9–12 May 1994.

[13] J.-S. Coron and D. Naccache. Boneh et al's k-element aggregate extraction assumption is equivalent to the Diffie-Hellman assumption. In C. Laih, editor, *Advances in Cryptology—ASIACRYPT-2003*, Taipei, Taiwan, Nov 30 – Dec 4, 2003. Springer-Verlag.

[14] Y. Desmedt, editor. *6th International Workshop on Practice and Theory in Public Key Cryptosystems — PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*. Springer-Verlag, Jan. 2003.

[15] Y. Dodis. Efficient construction of (distributed) verifiable random functions. In Desmedt [14].

[16] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In Y. Zheng, editor, *Advances in Cryptology—ASIACRYPT-2002*, volume 2501 of *Lecture Notes in Computer Science*, Queenstown, New Zealand, 1–5 Dec. 2002. Springer-Verlag.

[17] A. Joux. A one-round protocol for tripartite Diffie-Hellman. In *ANTS-IV Conference*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Spring-Verlag, 2000.

[18] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. IACR E-print Archive. Available from `http://eprint.iacr.org/2001/003/`, 2001.

[19] N. Koblitz. *A Course in Number Theory and Cryptography (second edition).* Springer Verlag, New York, NY, 1994.

[20] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Computer Science. Springer-Verlag, 18–22 Aug. 2002.

[21] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. In *Eighth ACM Conference on Computer and Communication Security*, pages 245–254. ACM, Nov. 5–8 2001.

[22] J. M. Park, E. Chong, H. Siegel, and I. Ray. Constructing fair exchange protocols for E-commerce via distributed computation of RSA signatures. In *22-th Annual ACM Symp. on Principles of Distributed Computing*, pages 172–181, 13–16 July 2003.