Probabilistic Counting of Large Digital Signature Collections

Markus G. Kuhn[*]

University of Cambridge
Computer Laboratory
Pembroke Street
Cambridge CB2 3QG
United Kingdom
mgk25@cl.cam.ac.uk

Abstract

A large number of people digitally sign the same document. The signature collectors want to use only a small amount of memory to demonstrate to any third party approximately how many persons have signed it. The scheme described in this paper uses a non-uniform secure hash function to select a small subset of signatures that the collectors store. The size of this subset becomes a verifiable estimate for the logarithm of the number of signers.

Applications for this scheme range from Web page metering, through ranking mechanisms for electronic discussion systems, to the distributed verifiable and scalable delegation of power in jointly-administrated systems.

1 Introduction

Signature collection has a long tradition as a means of documenting public support and plays an important rôle in many political systems. Candidates for political positions often have to collect signatures from voters as an entry qualification for an election. Some constitutions allow voters to initiate a vote by collecting a sufficient number of signatures. Civil rights organizations frequently use signature collection to demonstrate broad public support for their agendas.

Signature collection can be a valuable mechanism for decision-making arrangements that involve a large number of participants. It differs in important aspects from voting, which can consume significant up-front resources, since every entitled voter has to be informed early enough about the opportunity to vote. The outcome of a vote only represents the distribution of opinions sufficiently well when enough voters participate.

In a signature collection on the other hand, the signers can only decide to either support a proposal by signing a prepared document or to abstain from the procedure entirely. The result of a signature collection thus explicitly indicates only a lower bound for the absolute number of supporters for the proposal and it provides no useful information on how many people rejected it. Signature collection therefore makes sense only in situations where it is in the interest of the originators of the proposal—and those who collect signatures for it—to get as many signatures as possible.

Signature collection can be performed in a meaningful way without activating potentially expensive mechanisms for inviting every entitled participant. While voting requires an agreed official channel for announcing the start of the procedure and the details of the choices, in a signature collection, the responsibility for any necessary announcements can be left to the initiators of a proposal.

A centralized voting announcement mechanism could become subject to frequent frivolous abuse. Its activation must therefore be protected from malicious participants, who could try to delay further votes by overusing their right to initiate one. Signature collection mechanisms on the other hand can be started locally without large-scale announcements. They therefore make a much less attractive target for denial-of-service attacks.

Signature collections can potentially become very large, involving many millions of signers. The goal of voting is to identify a majority choice and therefore every single vote could become critical in deciding the outcome. On the other hand, the goal of a signature collection is only to demonstrate that some significant number of participants supports a pro-

[*] Supported by a European Commission Marie Curie training grant

Proceedings of the 9th USENIX Security Symposium, Denver, Colorado, USA, August 14–17, 2000, USENIX Association, pp. 73–83, ISBN 1-880446-18-9.

posal, and therefore a verifiable order-of-magnitude estimate for the number of signers is what matters rather than the precise count. The presentation of for example roughly a million supporting signatures can be an important demonstration of popularity of an idea, even if there is a 25% uncertainty with regard to the precise number of valid signatures.

The result of a conventional signature collection is presented by the collectors as boxes of handwritten paper lists. This form allows anyone to identify and verify a few selected sample signers, but it usually remains prohibitively expensive to create a database of all signers. Signed paper lists thus remain a widely accepted compromise between verifiability and privacy concerns.

In this paper, we propose a new technique for conducting the cryptographic equivalent of signature collection and outline some potential applications. Our technique features the following properties, the first three of which are shared with a handwritten signature collection:

- A small sample of the signers will become identifiable such that the correctness of the signature collection can be verified by contacting them. However, it is not necessary to provide a complete database of all signers in order to make the claimed number of supporting signatures verifiable.
- Only the order-of-magnitude number of valid signatures is practically verifiable via sampling and not the precise number.
- The result of a signature collection can be sent as a single protocol data unit to verifiers and no further interactions between the verifiers and the collectors will be necessary to complete the verification. This makes the scheme useful for store-and-forward delivery of the result.
- Many signature collectors can operate independently and can merge their results at a later time without risking multiple collection of the same signature by different collectors to influence the result.
- The verifiable result of a large-scale digital signature collection fits into a file not much larger than a few tens of kilobytes, which can be efficiently transmitted to and verified by many different parties.
- Signature collectors need only a small amount of storage capacity, just large enough to hold the compact final result that is provided to the verifiers.

2 Application Examples

In addition to the classical political applications, the digital equivalent of signature collection has also numerous interesting new uses in distributed systems. Some examples include:

2.1 Web Page Metering

Advertising companies sponsor the providers of free online content and services in exchange for presence on popular Web pages. The sponsors want evidence for the popularity of the points-of-visibility that they have purchased. A discussion of the requirements and some proposed protocols for Web page metering can be found in Naor and Pinkas [1].

If every user of the sponsored Web content automatically signed a *digital guest book* by submitting a message that confirms that this user was indeed interested in the presented content, then the content provider could easily prove the popularity of her pages to the sponsors. Compared to the secret-sharing based metering scheme suggested in [1], the signature collection scheme presented in the next section has the advantage that the approximate number of users does not have to be known in advance to select the metering parameters, and users do not have to fetch secret shares from a trusted third party before using a Web page. They only have to be part of a suitable public-key infrastructure, and the certificates do not even have to bind their web-metering keys to any cleartext name if their real identity must remain secret from both the content provider and the sponsor.

2.2 TV Rating and Opinion Polling

A closely related application field is counting the viewers of a TV channel or interactive opinion gathering via TV networks with return channels. Advertisement customers and other observers might not entirely trust the network operators to provide accurate viewer or voter counts.

Set-top boxes could implement a mechanism to allow TV viewers to participate in signature collections, the result of which could then be verified directly by anyone using the public keys of the set-top box (or conditional-access module) manufacturers, which would here also act as key certification authorities. Signature lists could be collected into a compact representation in local and regional network nodes, so as to avoid communication bottlenecks around a central collection point.

2.3 Newsgroup Contributor Ranking

In open electronic discussion forums such as Internet newsgroups or mailing lists, the quality of contributions varies greatly. Long-term participants in such forums could establish a cryptographic credibility record by collecting signatures from readers who considered their past contributions valuable. The size of these signature lists could be queried by new participants to get an initial estimate of the experience and signal-to-noise ratio of a contributor.

With the digital signature collection scheme presented here, such a ranking mechanism can be implemented without any additional trusted third party beyond the certification authorities that are required as part of any public-key infrastructure. If the signature collection result can be represented compactly, it can be distributed easily over the newsgroup server network, allowing every reader to verify the validity of the signature count efficiently and independently without relying on the integrity of distribution servers.

2.4 Delegation of Power and Joint Administration

It might be undesirable to have replicated repositories for public software and documents as well as discussion groups under the central control of a single organization. An interesting alternative would be a system design in which all participants can directly determine which individuals they would like to see authorized to perform administrative tasks, especially if there were an efficient way in which each storage server can independently verify such a direct authorization of an individual by a large user base.

Such systems could be set up so that owning a long list of recent supporting signatures from other participants would allow an individual directly to perform privileged operations like acting as a moderator with the right to clean-up inappropriate publications ("spam", insults, piracy, etc.). Here, the signature list becomes a way in which a large number of participants can delegate administrative power (usually with suitable time and scope restrictions) to individuals.

The same could be achieved with a voting mechanism. However, this would require common trust in some central voting server that determines and certifies the majority winners of this process. The compact representation of the verifiable result gives the signature collection approach robustness against the failure or compromise of other servers. Every server on which the administrative actions will be executed can independently and directly verify the public support for the moderators.

In all the previous applications, a trivial approach would be that the collector of signatures just stores all v collected signatures and hands them over to anyone who wants to verify that at least v signatures have indeed been collected. This would require $O(v)$ storage space, transmission time, and verification time, which can be prohibitive if the number v of collected signatures is large ($\gg 10^4$).

We therefore present a new technique that allows us to collect signatures and prove a probabilistic lower bound for their number to others using only $O(\log v)$ of storage space, transmission time, and verification effort.

3 A Probabilistic Representation of Signature Collections

The basic idea of this scheme is the representation of a complete collection of signatures by storing only a small set of sample signatures. The more signatures there have already been collected and the more signatures there are already in the sample set, the less likely it will be that the collector is allowed to add another collected signature to this sample set.

We have to agree in advance on the selection process for signatures that are allowed to get into the sample. This selection process will not be under the full control of the collector. It is designed such that the sample size grows logarithmically with the number of collected signatures. The size of the sample becomes this way a probabilistic indicator for the logarithm of the number v of collected signatures. An inspection of the $O(\log v)$ sample signatures alone allows us then to verify that the sampling process was performed properly and that at least around v signatures must have been given to the collector.

We assume that the following public-key infrastructure is in place:

Every person A who is entitled to participate in the planned type of signature collection is in possession of a private signing key K_A and a public verification key K'_A for some deterministic digital signature scheme. Each such person A has also received a certificate C_A that confirms for K'_A that the corresponding K_A is the only secret key that is available for A to be used in the collection. It is the responsibility of the certification authority that generated C_A to ensure that it is very difficult for anyone to obtain simultaneously valid certificates for more than one single signing key and that it is very difficult for not-entitled persons to get such a certificate.

The signature scheme under which the K_A and K'_A are used must be deterministic. By this we mean that for each message M, person A can only generate one single signature value that verifies under K'_A. The signature must not contain any freely selectable entropy. For instance using RSA signatures in the form of the `RSASSA-PKCS1-v1_5` scheme [2] and a fixed message digest function (say SHA-1) would be suitable. Using for example RSA with the PSS scheme [3] or DSS [4] to generate the signature, or leaving the signer a choice of multiple algorithms, must not be allowed. Otherwise a signing key owner could generate many different valid signatures for the same message M, which could allow her to get her signatures counted several times in the collection.

The signature collection is performed as follows:

1. The signature collector distributes to all potentially interested signers A the proposed message M together with an indication of which types of public-key certificates are accepted in this collection.

2. If person A decides not to sign M, she either does nothing or just confirms that she has received the proposal and is not interested in supporting it.

3. If A supports the proposal M, she generates the signature $s_A = \text{sign}(K_A, h(M))$ by applying her signing key K_A on a message digest of M, as required by the signature scheme. The resulting signature s_A is submitted together with the verification key K'_A and the certificate C_A to the signature collector.

4. The collector has a storage space consisting of n slots, each of which can save one signature, as well as the corresponding public key and certificate (or a pointer to it in some directory). The collector knows the verification keys for all certificate types that are allowed to be used.

5. The collector determines for each newly received signature the *slot position*

$$t = \varphi(H(s_A)),$$

where H is a uniformly distributed secure hash function that maps signatures onto binary words in the range 0 to $2^l - 1$, and

$$\varphi : \{0, \ldots, 2^l - 1\} \to \{1, \ldots, n\}$$

is a function that maps a uniformly distributed hash value onto one out of n non-uniformly distributed slot numbers (typical practical values are $l = 64$ and $100 < n < 5000$). A precise construction for φ will be suggested below.

6. If the collector has already stored a signature in slot number t, then the newly received signature will simply be discarded (see also section 4.4 for a better approach). Otherwise, the collector verifies that the received certificate C_A and the verification key K'_A are valid and that s_A is valid under K'_A for message M. If so, he will store s_A, K'_A, and C_A in slot number t.

7. Many collectors can be active in the same signature collection at the same time. The merged collection result will contain one sample signature for each slot for which at least one of the collectors has received a signature. If the same signature is submitted multiple times or to different collectors, this will not influence the number of filled slots. The same signature will always fall into the same slot with each collector and will therefore appear at most once in the end result.

It is intuitively clear that to find a signature whose hash has twenty leading zeros, we need to look at about 2^{20}, or a million, signatures. In what follows, we show how to count more precisely based on this general idea.

Let u be the number of slots that the collector has been able to fill with verifiable signatures. This number will be used to estimate the number v of valid distinct signatures that have been collected. The number u can be verified quickly by anyone who has reliable access to the public verification keys of the certification authorities, because for every filled slot number, one sample signature and the associated certificate has been kept as a proof that at least one signature for this slot was collected.

We describe the number of collected signatures using the discrete random variable V. The discrete random variables U_i with $1 \leq i \leq n$ describe the number of filled slots with slot numbers in the range 1 to i and $U = U_n$ shall be the random variable describing the total number of signatures that the collector has stored. The dependency between U and V is determined by the function φ, which defines the probability distribution of the random variable T that assigns signatures to slot numbers.

Let $p_t = P(T = t)$ be the probability that a uniformly distributed l-bit input hash value is mapped by φ onto slot number t, that is

$$p_t = \frac{\left|\{w \mid 0 \le w < 2^l \wedge \varphi(w) = t\}\right|}{2^l}, \quad \sum_{t=1}^{n} p_t = 1.$$

Given a distribution of T in the form of p_t values, a suitable φ can be constructed as

$$\varphi(w) = \min\left\{t \,\middle|\, 1 \le t \le n \wedge 2^l \cdot \sum_{i=1}^{t} p_i > w\right\}$$

and easily implemented as a binary search in a stored table of the values

$$w_i = \left\lceil 2^l \cdot \sum_{t=1}^{i} p_t \right\rceil, \text{ for all } i \in \{1, \dots, n-1\}.$$

The word size l of the secure hash function H has to be selected sufficiently large such that $2^{-l} \ll p_t$ for all $1 \le t \le n$.

After v different valid signatures have been collected, the probability that slot t is still empty will be $(1-p_t)^v$, which we will abbreviate in the following as $q_t := (1-p_t)^v$.

The probability that after v different signatures have been collected, exactly u signatures are stored in the first $1 \le i \le n$ slots is

$$r_i(u,v) := P(U_i = u | V = v) = \sum_{\substack{F \subseteq \{1,\dots,i\} \\ |F| = u}} \left(\prod_{t \in F} (1 - q_t) \right) \left(\prod_{t \in \{1,\dots,i\} \setminus F} q_t \right).$$

The above function can in spite of its exponentially long sum be computed efficiently in $O(n^2)$ time using the following recursion over the number of slots:

$$r_i(u,v) = \begin{cases} r_{i-1}(u,v) \cdot q_i + \\ r_{i-1}(u-1,v) \cdot (1-q_i), \\ \qquad \text{if } 0 \le u \le i \text{ and } i > 0 \\ 1, \qquad \text{if } i = u = 0 \\ 0, \qquad \text{otherwise} \end{cases}$$

We can now determine the expected number of filled slots among the first i slots recursively as

$$\begin{aligned} \overline{u_i} &:= E[U_i|V=v] = \sum_{u=0}^{i} u \cdot r_i(u,v) \\ &= \sum_{u=0}^{i} u \cdot [r_{i-1}(u,v) \cdot q_i + \\ &\quad r_{i-1}(u-1,v) \cdot (1-q_i)] \\ &= q_i \left(\sum_{u=0}^{i-1} u \cdot r_{i-1}(u,v) + i \cdot r_{i-1}(i,v) \right) + \\ &\quad (1-q_i) \sum_{u=-1}^{i-1} (u+1) \cdot r_{i-1}(u,v) \\ &= q_i \cdot (\overline{u_{i-1}} + 0) + (1-q_i) \cdot \\ &\quad \left(\sum_{u=0}^{i-1} u \cdot r_{i-1}(u,v) + \sum_{u=0}^{i-1} r_{i-1}(u,v) \right) \\ &= q_i \cdot \overline{u_{i-1}} + (1-q_i) \cdot (\overline{u_{i-1}} + 1) \\ &= \overline{u_{i-1}} + 1 - q_i, \end{aligned}$$

and therefore

$$\overline{u_i} = E[U_i|V=v] = \sum_{t=1}^{i} (1 - (1-p_t)^v).$$

We can similarly calculate the average square of the number of filled slots among the first i slots recursively as

$$\begin{aligned} \overline{u_i^2} &:= E[U_i^2|V=v] = \sum_{u=0}^{i} u^2 \cdot r_i(u,v) \\ &= q_i \left(\sum_{u=0}^{i-1} u^2 \cdot r_{i-1}(u,v) + i^2 \cdot r_{i-1}(i,v) \right) + \\ &\quad (1-q_i) \sum_{u=-1}^{i-1} (u+1)^2 \cdot r_{i-1}(u,v) \\ &= q_i \cdot (\overline{u_{i-1}^2} + 0) + \\ &\quad (1-q_i) \cdot (\overline{u_{i-1}^2} + 2\overline{u_{i-1}} + 1) \\ &= \overline{u_{i-1}^2} + 2\overline{u_{i-1}} + 1 - q_i \cdot (2\overline{u_{i-1}} + 1) \end{aligned}$$

and therefore

$$\overline{u_i^2} = E[U_i^2|V=v] = \sum_{t=1}^{i} (2\overline{u_{t-1}} + 1) \cdot (1 - (1-p_t)^v).$$

This way, we can determine the variance of the number of filled slots as

$$\begin{aligned} &\operatorname{Var}[U|V=v] = \sigma^2 = E[(U - E[U])^2|V=v] \\ &\quad = \ E[U^2|V=v] - E^2[U|V=v] = \overline{u_n^2} - \overline{u_n}^2. \end{aligned}$$

3.1 Selecting Slot Probabilities

Ideally, we should aim to choose the slot selection probabilities p_t and the corresponding φ in a way such that we get

$$E[U|V=v] \approx a \ln(v+1)$$

for some scaling factor a. Then $e^{\frac{u}{a}} - 1$ would be the corresponding estimated number of collected unique

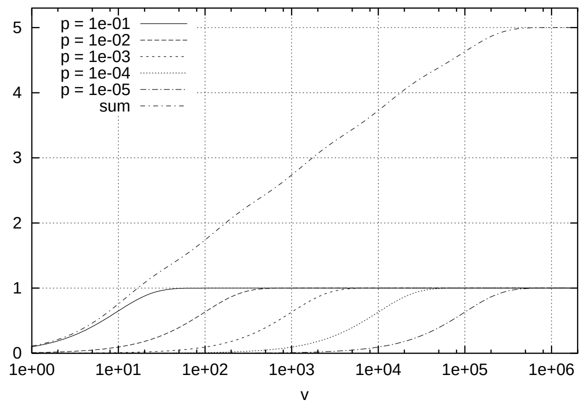

Figure 1: In this graph, we can see the function $1-(1-p)^v$ with the five parameter values $p = 10^{-1}, 10^{-2}, \ldots, 10^{-5}$ on a logarithmic scale for v, as well as the sum of these five curves. The $1-(1-p)^v$ graphs form smoothed unit steps near $1/p$ and the sum approximates a straight line (i.e., a logarithm) in the range 10^1 to 10^5.

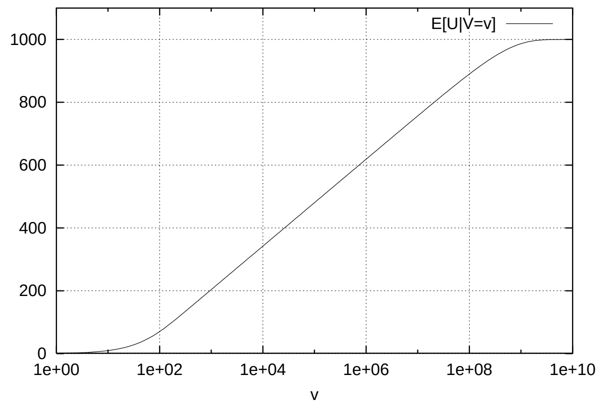

Figure 2: This graph shows the expected number of filled slots $E[U|V=v]$ over the number of collected signatures v for geometrically distributed slot selection probabilities p_t with parameters $n = 1000$ and $\beta = 0.9835$, selected such that up to $\hat{v} = 10^9$ signatures can be counted without danger of saturating the scheme.

valid signatures. The factor a determines the size of the sample set, and therefore the resolution and relative error of the estimated size of the collection. A logarithmic mapping between u and v corresponds to a constant relative error of the estimate of v over the entire range.

A good practical choice for φ turns out to be a geometric distribution of the form

$$p_t = \alpha\beta^t \tag{1}$$

with

$$\alpha = \left(\sum_{t=1}^{n} \beta^t\right)^{-1} = \frac{1-\beta}{\beta-\beta^{n+1}} \quad \text{and} \quad 0 < \beta < 1.$$

The reason that a geometric distribution leads to the desired result becomes intuitively clear as follows. We look at the graph of the function $1-(1-p)^v$ as we vary parameter p using values of a geometric series such as $p = 10^{-1}, 10^{-2}, \ldots, 10^{-5}$. These graphs are shown in Fig. 1, where v is plotted on a logarithmic scale. We see that this term results in a smooth unit-step function, and the location of the step coincides roughly with $1/p$, because if we solve $1-(1-p)^v = 1-e^{-1} \approx 0.63$ for v, we get

$$v = \frac{-1}{\ln(1-p)} \approx \frac{1}{p} \qquad \text{for } 0 < p \ll 1.$$

Since $E[U|V=v]$ is a sum of terms $1-(1-p)^v$, we can approximate on a logarithmic scale for v a straight line using equidistant values for $1/p$. This is demonstrated by the graph of the sum of the five smooth unit-step functions that is also shown in Fig. 1. It forms a nearly straight line in the range 10^1 to 10^5. This way, a geometric series of parameters p_t has resulted in $E[U|V=v]$ becoming an approximation for a logarithm of v.

Figure 2 shows $E[U|V=v]$ with a more practical set of parameters. In this example we use $n = 1000$ slots and we want to be able to handle up to $\hat{v} = 10^9$ signatures, for which $\beta = 0.9835$ is a good choice. With these parameters, the average relative error will be around 10% and therefore collection sizes that are less than a factor of two apart can still be separated very reliably. Using this parameter set, Fig. 3 shows for a number of collection sizes v the distribution $P(U=u|V=v)$. If a higher resolution is desired, the number n of slots has to be increased and β has to be adjusted accordingly.

If the desired number of slots n and the maximum expected signature count $\hat{v}$ are given, a suitable value for β fulfills the equation

$$1-(1-p_n)^{\hat{v}} = 1-e^{-1} \approx 0.63$$

such that the last slot n will be filled with a probability of a bit over one half after $\hat{v}$ signatures have been collected. The equation can be solved for β numerically using a binary search. Figure 4 shows for the practically useful ranges of n and $\hat{v}$ the corresponding β values and can be used to select a suitable β graphically.

With the geometric series of slot probabilities from (1), the probability densities $P(U=u|V=v)$

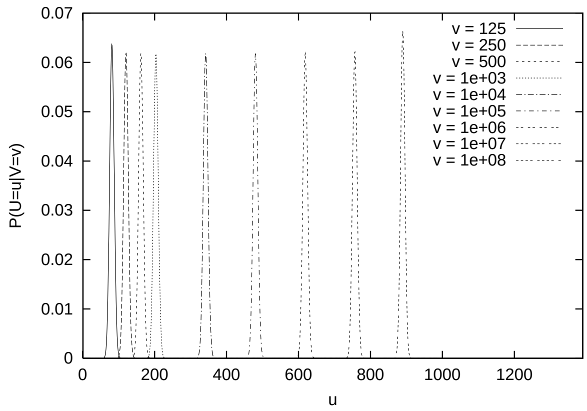

Figure 3: In this graph, the probability density $P(U = u|V = v)$ is shown for a number of signature collection sizes v. The parameters n and β are the same as in Fig. 2.

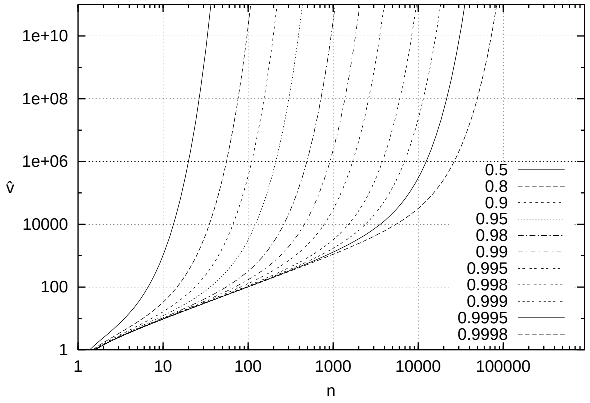

Figure 4: Using this graph, a suitable base β can be selected for a given number of slots n and a maximum expected signature count $\hat{v}$.

can be approximated by a Gaussian normal distribution:

$$P(U = u|V = v) \approx \frac{e^{-\frac{(u-\overline{u_n})^2}{2\overline{u_n^2}-2\overline{u_n}^2}}}{\sqrt{2\pi\left(\overline{u_n^2}-\overline{u_n}^2\right)}}$$

This works well at least as long as the variance is larger than one and the expected value is at least several standard deviations away from 0 and n, otherwise boundary effects make the shape of the distribution noticeably asymmetric.

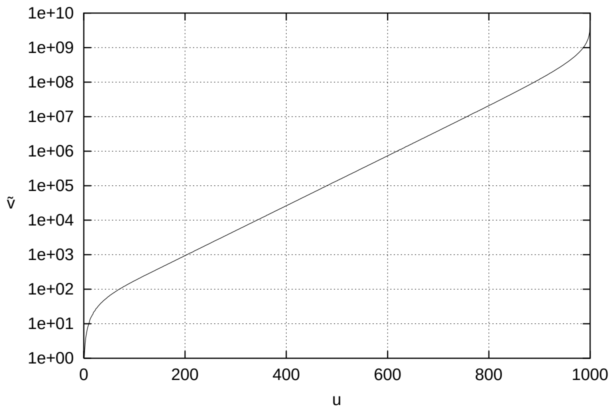

Figure 5: For the same parameters n and β as in Fig. 2, this graph shows the function $\tilde{v}$ that provides the maximum-likelihood estimate of v for a given u.

3.2 Estimating the Collection Result

We have seen how we can efficiently determine the distribution $P(U = u|V = v)$, but in order to interpret the given outcome u of a probabilistic signature collection, we are more interested in the Bayesian distribution

$$P(V = v|U = u) = \frac{P(U = u|V = v) \cdot P(V = v)}{\sum_{v'} P(U = u|V = v') \cdot P(V = v')}$$

that tells us how probable every possible number of signatures v was, given the outcome u. Unfortunately, we cannot practically determine this quantity unless we have a reasonable estimate for the distribution of V.

Therefore, the best practical estimate for v given a certain u will be the maximum-likelihood choice, i.e. the v for which $P(U = u|V = v)$ is maximal. Given the p_t values, it is possible to precalculate a function

$$\tilde{v} : \{0, \ldots, n\} \to \mathbb{N}$$

with $\tilde{v}(0) = 0$ and

$$P(U = u|V = \tilde{v}(u)) \geq P(U = u|V = v)$$

for all $u, v \in \mathbb{N}$ with $0 \leq u < n$, $0 \leq v$ and $P(U = n-1|V = v) > P(U = n|V = v)$. The value $\tilde{v}(n)$ remains undefined, because if all slots are filled, the only estimate about v that we can make is that $v > \tilde{v}(n-1)$, a case which should be avoided in applications by selecting a generously large value of $\hat{v}$.

For practical purposes, it is usually sufficient to determine $P(U = u|V = v)$ for v increasing in steps of around 5% and then tabulate in $\tilde{v}(u)$ that v that resulted in the largest $P(U = u|V = v)$. Figure 5 shows the result for our example parameter selection.

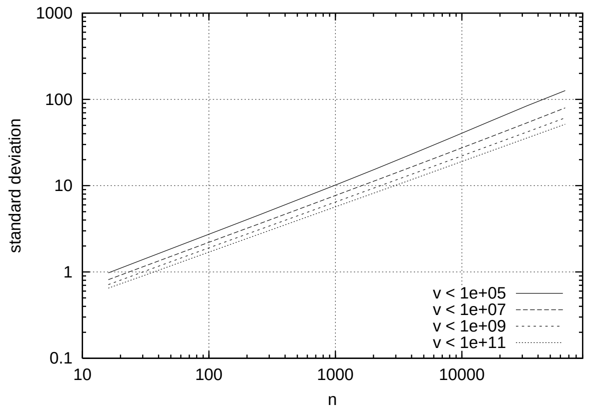

Figure 6: This graph shows for various n and $\hat{v}$ values the standard deviation $E[(U - E[U])^2]^{\frac{1}{2}}$ when $V = \tilde{v}(\frac{n}{2})$. It grows proportional to $\sqrt{n}$.

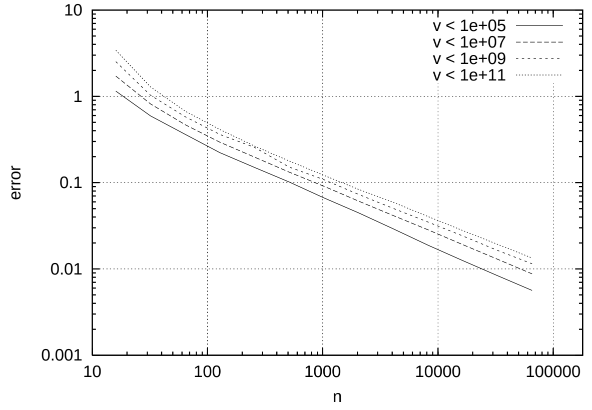

Figure 7: The root-mean-square of the relative error $(E[(\tilde{v}(U) - V)^2/V^2]^{\frac{1}{2}}$ for $V = \tilde{v}(\frac{n}{2}))$, which this graph shows for various n and $\hat{v}$ values, falls proportional to $1/\sqrt{n}$.

Alternatively, $\tilde{v}$ can also be determined simply by inverting $E(U|V = v)$, which is practically equivalent since the expected value happens to coincide well with the peak of $P(U = u|V = v)$ (see Fig. 5 compared to Fig. 2).

Figure 6 shows how the standard deviation of U depends for a fixed v on the parameters n and $\hat{v}$. It depends only slightly on $\hat{v}$ and grows proportional to $\sqrt{n}$. The root-mean-square of the relative error of the estimate $\tilde{v}$ develops accordingly proportional to $1/\sqrt{n}$ (Fig. 7).

Instead of just looking at the number u of collected signatures, the verifiers can also base their maximum-likelihood estimate of v on the set $F \subseteq \{1, \ldots, n\}$ of slot positions that have been filled. They can then make their estimate using the function $\tilde{v}(F)$ that returns that value of v which maximizes

$$\prod_{t \in F} (1 - (1 - p_t)^v) \cdot \prod_{t \notin F} (1 - p_t)^v.$$

This approach allows the verifiers to utilize all available information, but it also makes it impractical to precompute the function $\tilde{v}$ as a simple table.

Verifiers do not necessarily have to check the content of all slots. If many filled slots with low p_t have been verified, all those slots with orders-of-magnitude larger p_t will most likely be full as well. This allows the result to be represented even more compactly. The performance of the scheme could also be improved by using several slot mapping functions φ of different granularity. When a collector's storage space becomes full, he switches to the next φ and merges existing slots accordingly.

4 Security Considerations

4.1 Trust Placed in Collectors

Collectors can easily discard signatures and can therefore make the number of submitted signatures look smaller than it actually was. As a consequence, the result of a signature collection can only be meaningful if signatures were collected by someone with a genuine incentive to collect as many signatures as possible.

It is also worth noting that participating signers have no guarantee that signature collectors store only a maximum of n signatures. While signature collectors are only required to store and present $u = O(\log v)$ signatures to the verifiers, nothing prevents them from creating a record of all v signatures. The privacy protection that signature collection offers thus protects only the signer from the verifier, but not the signer from the signature collector.

4.2 Bribery

If either collectors or the signing key holders know in advance which signing keys are able to fill a slot, this might with some applications lead to the creation of a market for those sample signatures that fill slot positions with low p_t values. The resulting risk is that instead of convincing $O(v)$ people of the value of proposition M, the collectors find it easier to secretly purchase suitable sample signatures from $O(\log v)$ selected signers to achieve the same result.

The presented scheme assigns the slot number $t = \varphi(H(s_A))$ based on participant A's signature $s_A = \text{sign}(K_A, h(M))$ of document M. It is the very nature of a digital signature that its value is not predictable by anyone except the holder of the signing key. This way, the signature collectors have no way of identifying in advance those signing key holders who could provide them with the valuable sample signatures that will fall into slots with low p_t and that therefore represent a large number of signatures. If the slot number were instead assigned based on the signer's identity and not her signature of M, then the signature collector could identify in advance those few signers whose signatures they need to fill the low-probability slots. Strong incentives could be offered to these few individuals to sign the proposal.

If φ and H are publicly known, then all potential signers can check whether their signature would fall into a slot with a low p_t value. They could contact the signature collectors secretly and could offer to sell their more valuable signatures.

In applications where this second kind of bribery is of concern, a secret nonce N only known to the collectors should be concatenated with the signature in the calculation of the slot position

$$t = \varphi(H(N \parallel s_A)).$$

This will prevent signers from determining whether their signature could fill an unlikely slot. The nonce N will be published *after* the signature collection process has finished, as this parameter has to be accessible to the verifiers. Every sample signature that is stored in a slot position has to be registered at a timestamping service *before* the publication of N. The early leakage of N into the public only becomes a problem if the number of signers who hear about N during the collection is of an order of magnitude comparable to the number of signatures that the collectors will later claim to have obtained. In this case, it is also very likely that someone opposing the proposal will learn N *before* the collection closes. This person can later prove the information leakage by registering N immediately at a time stamping service, and thus gains strong evidence for early public knowledge of N, which can then invalidate the entire signature collection.

4.3 Collusion of Signers

Another threat that has to be taken into account is that a group of signing key holders could collude with the signature collectors before the start of the collection. They could generate a very large number of candidate documents M with slight variations until they find one M for which the signing keys of the members of that group can be used to fill an unusually high number of slots. They then start a signature collection campaign with this carefully phrased proposal M that guarantees their own secret keys a high impact.

If c signers collude with the signature collectors, then the probability that an arbitrary message M will allow these c signers to fill at least w slots is

$$\sum_{i=w}^{n} P(U = i | V = c).$$

In order to find with sufficient probability (say $1 - e^{-1} \approx 63\%$) such a message M that fills at least w slots, they would have to generate at least

$$\begin{aligned} m &= \frac{-1}{\ln\left(1 - \sum_{i=w}^{n} P(U = i | V = c)\right)} \\ &\approx \frac{1}{\sum_{i=w}^{n} P(U = i | V = c)} \end{aligned}$$

candidate messages M and then have to calculate all cm signatures on these to determine the slot position of each signing key and message pair. This should become computationally very impractical when $cm \gg 10^{20}$.

Figure 8 shows the example distributions from Fig. 3 on a logarithmic scale. Like a Gaussian normal distribution, the $P(U = u|V = v)$ curves have a parabolic shape on a logarithmic scale and fall off rapidly. It is unlikely that a group of c signers who collude with the signature collectors can increase the number w of illegitimate sample signatures to more than nine standard deviations above $E[U|V = c]$.

In addition, if the true number of collected signatures v is larger than $\tilde{v}(w)$, then the collusion is unlikely to have any significant effect on the final $\tilde{v}(u)$, because most of the w slots filled by keys of the colluding group would have been filled anyway by legitimate signatures from non-colluding signers, and then the w illegitimate sample signatures do not contribute any more to the estimate $\tilde{v}(u)$.

A collusion could be prevented by requiring the signature collectors to announce M to some trusted party, which in return generates a nonce that has to be concatenated with M before the entire signature collection can start. This way, the signature collectors cannot carefully select M to maximize the impact of the colluding signers. For such a mechanism to be effective, however, the rate at which such announcements of M are allowed has to be limited, as otherwise, the announcement would not prevent the generation of a large number of candidate messages. Such a rate limitation, however, would defy the design goal of keeping the signature collection scheme free of central facilities and bottlenecks.

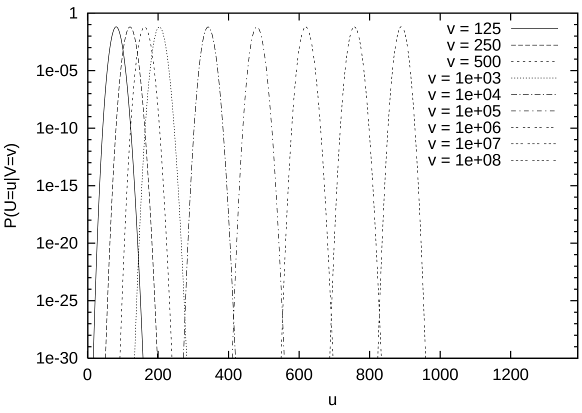

Figure 8: This graph is identical to Fig. 3 but shows the probabilities on a logarithmic scale.

4.4 Fair Sampling

An indication for a collusion of signers would be if an unusually large number of sample signatures came from signers who are likely to be affiliated with the collectors. This effect can also appear if only the first received signature for each slot position becomes the sample signature for that slot. It would not be unusual for signers in the social context of the collectors to submit their signatures earlier than most others and thus have a higher probability of ending up as sample signers.

For this reason—as well as in the interest of general fairness—it is advisable for the signature collectors to give every submitted signature an equal chance of becoming a sample. This can be achieved by keeping a counter g_t for every slot t, which indicates how many valid signatures have already been received for this slot. Whenever a new signature is received for slot t and is validated successfully, then g_t is increased by one, and the newly arrived signature will replace the previously stored one with probability $1/g_t$. This ensures that within a slot position, all submitted signatures have an equal probability of becoming the final sample, independent of the order in which they arrive.

This technique works because, if $k_i = 1/i$ is the probability for keeping the i-th arriving signature and n signatures are received in total, then the probability f_i with which the i-th signature becomes the final surviving sample is

$$\begin{aligned} f_i &= k_i \cdot \prod_{j=i+1}^{n} (1 - k_j) = \\ &= \frac{1}{i} \cdot \frac{i}{i+1} \cdot \frac{i+1}{i+2} \cdots \frac{n-1}{n} = \frac{1}{n} \end{aligned}$$

and therefore equal for all arrival positions i.

If signature collection results from the collectors $L_1, \ldots, L_d$ are merged, then every collector L_i provides for each slot t his submission count $g_{t,i}$. With probability $g_{t,i}/\sum_{j=1}^{d} g_{t,j}$ the sample signature in slot t of L_i will go into slot t of the merged result, and the submission count of this slot will be set to $\sum_{j=1}^{d} g_{t,j}$. This maintains the equal chance for all signature submissions to end up in the final result, provided that signers cooperate and do not submit signatures multiple times. Partial collection results should be merged only in a hierarchical way.

4.5 Cryptographic Assumptions

Signing messages that have been generated by others requires some diligence to avoid exposure to various attacks. The chosen signature scheme should certainly be robust against adaptive chosen message attacks. In addition, the syntax of the proposed message M should be sufficiently restricted to avoid that M can have any other uses then being interpreted as a proposal that was written for a signature collection.

It has been suggested to avoid signing messages without adding some entropy first [3], but the addition of random “salt” values seems not to be an applicable option against hypothetical future attacks in this application. The digital signature design community is therefore invited to consider signature collection based on probabilistic counting—apart from the well known subliminal-channel concerns of salt values—as yet another good reason for not giving up deterministic signature algorithms and to continue their design, evaluation, and standardization.

It is important to note that the presented scheme relies on a property of the utilized deterministic digital signature method that is not among those commonly studied in-depth. We assume that it is infeasible with the used signature scheme to generate an *ambiguous verification key* for which there is a practical way to generate many different signatures for the same document M that all verify correctly under this single public key[1]. An otherwise perfectly safe deterministic signature scheme might be unsuitable or at least require additional tests by the certification authority or the verifier and signature collectors on the public keys in order to keep individual signers from submitting many different seemingly valid signatures. If the chosen signature scheme allows the generation of ambiguous verification keys that

[1] With RSA, there exists exactly one signature s for $h(M)$ under public key (e, n) (with $s^e \bmod n = h(M)$), as long as $\gcd(e, \phi(n)) = 1$.

cannot be detected later, then the certification authority will have to generate the key pair and certify for the verification key not only the identity of the owner, but also that this key verifies only one single signature for any document.

5 Related Work

The general idea of *probabilistic counting* was probably first introduced by Morris [5], who used a small integer register to estimate how often an event had occurred. The number of events can be considerably larger than the maximum value that the register can hold, therefore the register is increased by one only with a certain probability when a new event occurs, and this probability is reduced exponentially as the register value increases. The register value becomes an estimate for the logarithm of the number of events. The logarithmic relationship keeps the relative error constant.

The probabilistic counting concept was later extended by Flajolet and Martin [6] to estimate the number of different values in a large database table. In their algorithm, every value is transformed using a hash function into a word. The position of the first 1-digit in this word is determined, and the corresponding bit is set in a bitmap. The position of the first zero in the resulting bitmap is used as an estimate for the logarithm of the number of different values that were processed this way. Multiple identical values are automatically discarded, as they would only set the same bit several times. The counting process can easily be distributed and the resulting bitmaps can be combined using a logical-*or* operation. The probability that the first 1-digit in a uniformly distributed word is the n-th digit is 2^{-n}, and the position of the first zero digit in the bitmap is therefore comparable to the value of Morris' register. This technique was later extended by Kirschenhofer, Prodinger and Szpankowski by using a histogram with saturation arithmetic instead of a bitmap to increase counting accuracy [7].

6 Conclusions

The cryptographic equivalent of signature collection can be implemented using a probabilistic counting technique that provides efficient verification of the result. We have developed and discussed a practical construction of such a process with application properties very similar to those of handwritten signature collections. The presented numerical analysis will help in the selection of suitable parameters. We have discussed how Web page metering, TV rating and opinion gathering, newsgroup contribution ranking, and joint administration concepts are among the possible application areas. This signature collection technique can lead to improvements in robustness, privacy protection, and efficiency, as long as the application designer fully understands the described security considerations. Perhaps this application idea will encourage the development of public-key infrastructures which support a new key and certificate type that provides the assurance that an individual cannot easily generate more than one verifiable signature for a message.

The author wants to thank Ross Anderson, David Wheeler and the referees for valuable comments.

References

[1] Moni Naor, Benny Pinkas: Secure and Efficient Metering. In Kaisa Nyberg (ed.): *Advances in Cryptology – EUROCRYPT '98*, LNCS 1403, pp. 576–590. Springer-Verlag, May/June 1998.

[2] Burt Kaliski, Jessica Staddon: PKCS #1: RSA Cryptography Standard – Version 2.0. RSA Laboratories, September 1998. `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/`

[3] Mihir Bellare, Phillip Rogaway: The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In Ueli Maurer (ed.): *Advances in Cryptology – EUROCRYPT '96*, LNCS 1070, pp. 399–416. Springer-Verlag, May 1998.

[4] Digital Signature Standard (DSS). Federal Information Processing Standards Publication FIPS PUB 186-1, U.S. Department of Commerce, December 1988.

[5] Robert Morris: Counting Large Numbers of Events in Small Registers. *Communications of the ACM*, Vol. 21, No. 10, pp. 840–842, October 1978.

[6] Philippe Flajolet, G. Nigel Martin: Probabilistic Counting Algorithms for Data Base Applications. *Journal of Computer and System Sciences*, Vol. 31, No. 2, pp. 182–209, October 1985.

[7] Peter Kirschenhofer, Helmut Prodinger, Wojciech Szpankowski: How to Count Quickly and Accurately: A Unified Analysis of Probabilistic Counting and Other Related Problems. In W. Kuich (ed.): *Automata, Languages and Programming*, 19th International Colloquium, pp. 211–222, Wien, Austria, 13–17 July 1992. Springer-Verlag.