8.7 Prediction Resistance and Backtracking Resistance

Comment [ebb1]: Page: 48
This section could be empty for now. However, the current text indicates a possible discussion (that may not be correct right now).

Each of the DRBGs specified in Section 10 has been designed to provide prediction resistance and backtracking resistance when observed from outside the DRBG boundary, given that the observer does not know the seed, or any key or state values.
Figure 6 depicts the sequence of DRBG states that result from a given seed. Some subset of bits from each state are used to generate pseudorandom bits upon request by a user. The following discussions will use the figure to explain backtracking and prediction resistance. Suppose that the user wants assurance that an adversary cannot determine the pseudorandom bits produced from $State_x$.

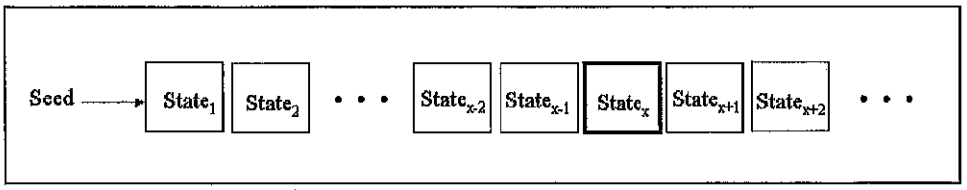

Figure 6: Sequence of DRBG States

Backtracking Resistance: When a DRBG provides backtracking resistance, an adversary is unable to determine the bits in $State_x$ if that adversary is able to determine the bits in any state occurring subsequent to $State_x$. That is, if $State_{x+1}$ (or any state after $State_{x+1}$) is compromised, the adversary is unable to "back up" the process to determine the bits in $State_x$. Each of the DRBGs in this Standard provide backtracking resistance, with the exception of the **SHA1_Hash_DRBG** specified in Section 10.1.2. When observed from within the DRBG boundary (i.e., the DRBG is observed as a glass box, and the advsersary can get the current state (e.g., $State_{x+1}$)), the previous states cannot be determined. Because the **SHA1_Hash_DRBG** does not provide backtracking resistance, this DRBG **should not** be used for new applications, only for existing applications where interoperability is required.
Prediction Resistance: When prediction resistance is provided, an adversary is unable to determine the bits in $State_x$ if that adversary is able to determine the bits in any state prior to $State_x$. That is, if $State_{x-1}$ (or any state prior to $State_{x-1}$) is compromised, the adversary is unable to generate the next bits in the process and so (ultimately) to determine the bits in $State_x$. Note that an adversary will normally be able to determine the next bits if prediction resistance is not provided because of the deterministic nature of the DRBG. When observed from within the DRBG boundary (that is, as a glass box where the current state (e.g., $State_{x-1}$) is known), prediction resistance may be provided for a DRBG by the insertion of sufficient additional entropy prior to generating pseudorandom bits; for example, by doing an explicit reseed. Sufficient entropy is defined as being at least equal to the amount of entropy required for the seed used to instantiate the DRBG at the desired security strength (i.e., *min-entropy* = **max** (128, *strength*); see Section 8.4, item 2). Providing the additional entropy prior to generating new pseudorandom bits (i.e., generating a new state) isolates the newly generated bits from prior bits generated by the DRBG (i.e., from prior states); knowledge of previously generated bits (e.g., obtained via

a compromise) does not allow the prediction of the new bits. In this Standard, the **SHA1_Hash_DRBG** has not been specified to provide prediction resistance.
Note that prediction resistance is not provided if the entropy is obtained in amounts that are less than required to support the desired security level. Inserting insufficient additional entropy is better than not inserting additional entropy at all, but the DRBG cannot provide prediction resistance.